DOS EXTRA Nr. 7 – 1989

DOS EXTRA

18,– DM 145 Ös 18,– sfr

Sonderausgabe DOS EXTRA

Nr. 7 1989

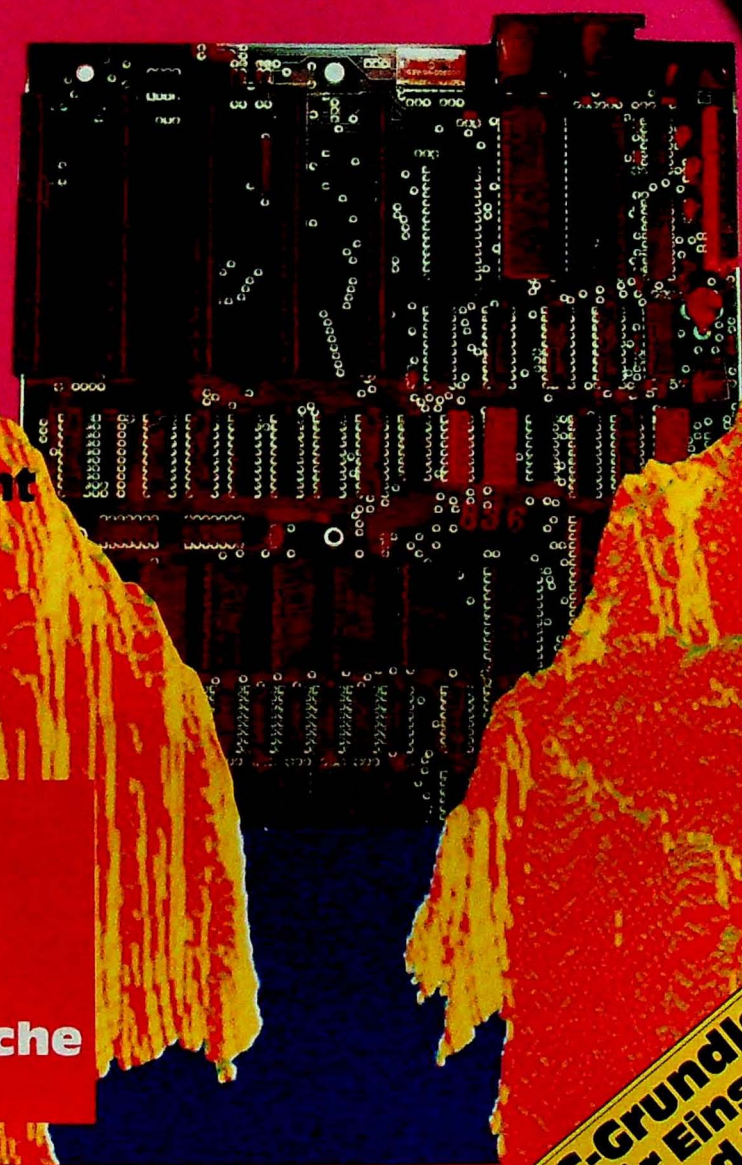
ISSN 0934 – 2842

DMV-Verlag

Eine Publikation der
DOS INTERNATIONAL

PC-Komponenten im Detail

- Bausteine
- Controller
- Schnittstellen



Utilities

- Sprite-Editor für animierte Grafik
- MS-DOS leichtgemacht mit Easy-DOS
- DOS-Packer

STARK

SBasic noch professioneller

- Benutzeroberfläche
- Präprozessor

C-Grundlagen für Einsteiger und Profis

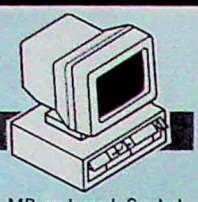
CPS

CPS Computertechnik GmbH
Marienstraße 16 • 3300 Braunschweig
Fax (05 31) 79 64 61 • BTX *20088 1490#

Großes Angebot an PC/XT-AT-kompatiblen Rechnern, Zubehör und Software !
Studentenrabatte geg. Nachweis a. Anfrage !

Tel. (05 31) 794087

CPS XT und AT



CPS XT I
12 MHz, 256 KB RAM bis 1 MB on board. Sockel für Co-Prozessor, Dualgrafikk. (Herc. & Color), Multi I/O, 1 Drive 5,25" 360k, 20 MB Festplatte ser./par., Game, Clock, 14" Flat Screen Monitor s/w, Best.Nr. 00550 2000,-

CPS AT I (siehe Test DOS 1/89)
80286-12 CPU 8/10/12 MHz (16 MHz Landmark), 512 KB RAM bis 4MB on board. 0-Wait, Sockel für Co-Prozessor, EMS 4.0, Phoenix Bios 64K ROM mit Setup-Funktion, Uhr u. Kalender, 1 ser./1 par./Game-Port, Dualgrafikkarte (Herc. & Color) 20 MB Festpl. 38ms, 1 Drive 5,25" 1,2 MB und 1 Drive 3,5" 720K/1,44MB, Cherry Tastatur m. sep. Cursorbl., 14" Flat Screen Monitor s/w oder amber, Best.Nr. 00560 3540,-

CPS AT Suntac (siehe Testbericht MegaByte 1/89)
wie AT I jedoch Festplatte 28ms, Slimline Gehäuse, Best.Nr. 00570 3740,-

CPS AT II
80286-16 CPU 8/10/12/16 MHz (20 MHz Landmark), 512 KB RAM bis 4 MB on board. EMS 4.0, 0-Wait, sonst wie AT I, Best.Nr. 00580 4050,-

CPS AT III
80286-12 CPU 8/12 MHz (16 MHz Landmark), 512 KB RAM bis 8 MB on board. EMS 4.0, 0-Wait, AMI Bios, Page/Interleave Speicher Controll, sonst wie AT I, Best.Nr. 00590 3850,-

CPS AT 386-16
80386-16 CPU 16 MHz (20 MHz Landmark), 2 MB RAM bis 16 MB on board. 0-Wait, Sockel für Co-Prozessor. Phoenix BIOS 64 K ROM, Dual-Grafikk. (Herc. & Color), Uhr u. Kalender, 1 ser./1 par./Maus Port, 1 Drive 5,25" 1,2 MB und 1 Drive 3,5" 720K/1,44 MB, 20 MB Festplatte 28ms, 102 MF II Cherry Tastatur, MS-DOS 3.3 und GW-Basic, GEM 3.0, Q-Mouse, 14" Flat Screen Monitor s/w, Best. Nr. 00600 7470,-

Gerätekonfigurationen nach Ihren Wünschen veränderbar. Speichererweiterungen bitte Tagespreis anfragen.

18 Monate Garantie auf CPS Rechner!

Disketten

NN 2DD 3,5" 10 Stck. 20,80
NN 2DD 5,25" 10 Stck. 9,-

FARBBÄNDER

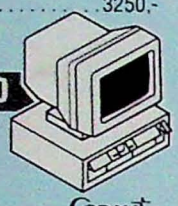
Mindestabnahme 3 Stück
STAR NL/NG/ND/NR-10, Stück 12,-
EPSON LX-800/LQ-500, Stück 11,-
PANASONIC KX-P, Stück 13,-

Nur solange der Vorrat reicht...

CPS 286/20
80286-10 CPU 8/10 MHz, 0-Wait, 512 KB RAM (1 MB optional), Sockel für Co-Prozessor, DUAL-Grafikk.(Herc. & Color), Uhr u. Kalender, 2 ser./2 par./Mouse-Port, 102 MF-II Tastatur, PHOENIX Bios 64 K ROM mit Setup-Funktion, 1 Drive 5,25" 1,2 MB, 20 MB Festplatte 65ms, MS-DOS 3.3 u. GW-Basic, GEM 3.0, Q-Mouse, 14" Flat Screen Monitor s/w, Best. Nr. 00610 3250,-

18 Monate Garantie!

Copam 80286-C-120

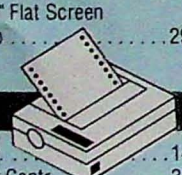


80286 12 MHz (16 MHz Landmark), 0-Wait, 512 KB RAM, Sockel 80287, Hercules komp. MGA/CGA Grafikkarte, Q-Mouse, Uhr u. Kalender, 2 ser./2 par. Port, Mouse-Port, PHOENIX Bios 64K ROM mit Setup-Funktion, 1 Drive 1,2 MB 5,25" 20 MB Festplatte 38ms, MF-II Tastatur mit 102 Tasten, Formschönes Gehäuse mit Betriebs-LED's, Turbo, Reset und Schlüsselschalter, 14" Flat Screen Monitor s/w mit Dreh- und Schwenkfuß, Software, MS-DOS 3.3 und GW-Basic, Handbuch, GEM 3.0, Best.Nr. 00620.
Garantie 18 Monate
Komplett-Preis 3480,-

Kenitec 286-12

80286-12 CPU 8/12 MHz (16 MHz Landmark), 512 KB RAM, (1 MB optional) 0-Wait, Sockel 80287, Herk. komp. Grafikk., LEGAL-Bios, 2 ser./1 par., 1 Lw 5,25" 1,2 MB, 20 MB Festplatte. 65ms, Uhr u. Kalender, 102 MF-II Tastatur, **Garantie 18 Monate**, 14" Flat Screen Monitor s/w, Best.Nr. 00630 2990,-

DRUCKER



OKI Microline 390 Centr. 1498,-
OKI Microline 393 Color Centr. 3079,-
NEC P 6 Plus Centr. 1798,-
NEC P 7 Plus Centr. 1985,-
NEC P 2200 Centr. 898,-
STAR LC 10 Centr. 615,-
EPSON LX 800 Centr. 549,-
EPSON LQ 500 Centr. 992,-
EPSON LQ 1050 Centr. 1950,-
Centronics Druckerkabel 18,-
Weitere Drucker auf Anfrage

Wir liefern nur mit dt. Handbuch, Seriennummer und Herstellergarantie !!!Drucker-Grauimporte mit engl. Handbuch, ohne Seriennummer, ohne Herstellergarantie sind bei uns ausgeschlossen.

Nutzen Sie unseren Bequem-Kauf-Kredit!

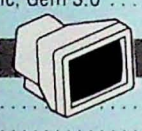
COMMODORE AMIGA/PC

AMIGA 500 950,-
AMIGA 2000 1980,-
AMIGA 500 + Mon. 1084 S 1520,-
AMIGA 2000 + 20 MB Harddisk SCSI 2840,-
PC/XT-Karte inkl. 5,25" LW, MS-DOS + GW-Basic 820,-
LW ext. AMIGA 500 3,5" 266,-
LW ext. AMIGA 2000 5,25" 419,-
LW int. AMIGA 2000 3,5" 220,-
RGB Commodore 1084 S 630,-
RGB Stereo Philips CM 8833 630,-
PC 10/III, 20 MB Festpl. MS-DOS 2200,-
PC 20/III, 20 MB Festpl., MS-DOS 2300,-

CPS AT 286-16 Tower

(siehe Testbericht FORMAT 5/89)
80286-16 CPU 8/10/12/16 MHz (20 MHz Landmark), 1 MB RAM bis 4 MB on board, 40 MB Festpl. 40ms, EMS, 0-Wait, Sockel für 80287, Phoenix Bios 64 K ROM mit Setup-Funktion, Uhr u. Kalender, 1 ser./1 par./Port, Herc. komp. Grafikk., 1 Drive 5,25" 1,2 MB und 1 Drive 3,5" 720K/1,44 MB, Cherry Tastatur m. sep. Cursorbl., 14" Flat Screen Monitor, Mouse RS 232, MS-DOS 3.3 und GW-Basic, Gem 3.0 5740,-

MONITORE



EGA Philips/Thomson 718,-
EIZO 8060s 1680,-
EIZO 9070s 2320,-
NEC MULTISYNC II 1500,-
NEC MULTISYNC GS 630,-
TTL 14" Flat Screen m. Fuß sw/amber 225,-
TTL 14" Flat Screen grün invers 272,-
weitere Monitore auf Anfrage

Die Personal Computer mit 2-Jahres-Garantie —VOR ORT.

ARCHE

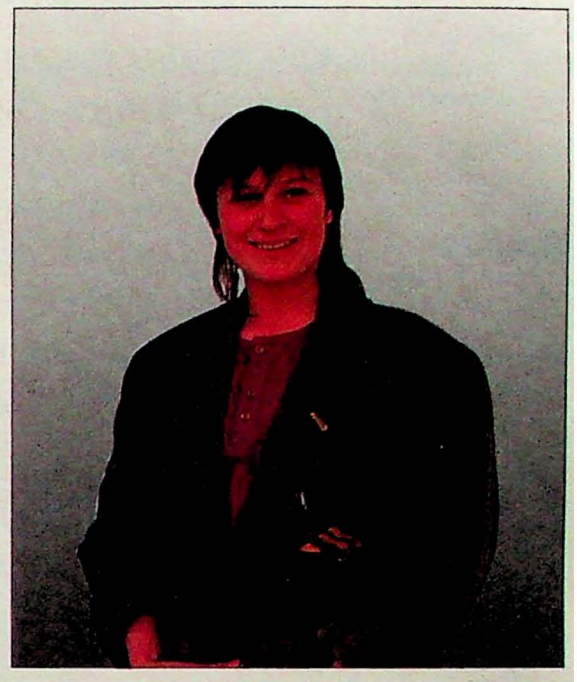
RIVAL 286/20
Intel 80286 mit 12/8 MHz, Hauptplatine mit 512 KB RAM, ausbaubar bis 16 MB (1MB auf Motherboard), Sockel für 80287 Co-Prozessor Award Bios, 6 x 16 bit und 2 x 8 bit Anschlüsse, 200 Watt Netzteil, Grafikkarte monochrom, Hercules komp., 2 ser./2 par. Schnittstellen, davon 1 auf Hercules-Karte, 102-er MF-Tastatur, AT-Gehäuse im professionellen ARCHE-Design, dt. Handbuch, 1 Floppy Laufwerk 5,25"/1,2 MB, 1 Festplatte 20 MB 4412,-

Sämtliche Angebote freibleibend. Zwischenverkauf vorbehalten. Wir liefern nur per UPS-Nachnahme in der Regel innerhalb 24 Stunden, ins Ausland nur gegen Vorkasse. In Ausnahmefällen kann es zu herstellerbedingten Lieferengpässen kommen. Fordern Sie unseren Gesamtkatalog gegen 3,- DM in Briefmarken.
Es gelten unsere allgemeinen Geschäftsbedingungen.

CPS **weil Preis & Leistung stimmen !**

Wir sind autorisierter Fachhändler für:

Copam Electronics Corp. · Tulip computers · ARCHE TECHNOLOGIES · Acer · Kenitec · EPSON · olivetti



Wie es Euch gefällt...

Es ist nicht immer ganz einfach für ein Sonderheft, also für unsere DOS Extra, die richtige Themenauswahl zusammenzustellen. Nach welchen Kriterien soll man bei der Themenwahl vorgehen, und wer soll angesprochen werden? Auf einen einfachen Nenner gebracht: Wir haben uns für diese Sonderheftreihe vorgenommen, alle Themen, die in der DOS International aus Platzgründen zu kurz kommen, aber doch auf ein großes Interesse seitens unserer Leser stoßen, aufzugreifen und vertieft und ausführlich zu behandeln.

An Themen mangelt es nicht, doch glücklicherweise gab uns Ihre rege Beteiligung an der Frageaktion der DOS Extra 6 Auskunft darüber, wo Ihre Hauptinteressen liegen. Ganz vorne liegen die Themen Basic, Pascal und C, Grafik, Hardware und Assembler, wobei dem Thema Basic mit der DOS Extra 6 bereits eine vollständige Ausgabe gewidmet wurde.

Für dieses Heft haben wir uns wieder für eine bunte Palette an Themen entschieden: Es geht um Grafik, um eine Expedition in die Innenwelt der PC-Hardware sowie um einen C-Kurs, womit – so hoffen wir – ein breites Spektrum verschiedenster Interessen abgedeckt ist. Doch was erwartet Sie nun im Detail?

Der Faszination der Computergrafik kann sich wohl kaum ein PC-Besitzer entziehen, gleichgültig, ob er selbst programmiert oder sich nur an Fractals oder ähnlich eindrucksvollen Beweisen der Grafikfähigkeiten seines Computers ergötzen will. Da unsere DOS-Extra-Sonderheftreihe jedoch insbesondere den aktiven PC-Anwender anspricht, gehen auch die Beiträge in diese Richtung. So informieren zwei Werkstätten über die Programmierung von CGA- und Hercules-Grafikkarte, und einen Sprite-Editor für den PC wünschen sich viele Grafikfans sicherlich schon lange: Hier ist er!

Für ganz Gründliche und Wißbegierige ist der Themenkreis »PC intern« gedacht, der für Sie die Hardware des PC im Detail »seziert«. Wie arbeiten die diversen Schnittstellen? Was erledigen die verschiedenen Controller? Wie funktioniert der Timerbaustein?

Diese und derlei Fragen mehr beantwortet eine ausgesuchte Reihe fundierter Beiträge.

Der dritte Themenkomplex ist der Trendsprache C gewidmet. Er ist nicht als Einsteigerkurs konzipiert, sondern seine Beiträge befassen sich in loser Folge mit »Spezialitäten« wie Zeigern, Operatoren, Vektoren oder dem Präprozessor der Programmiersprache C. Es wird dargelegt, warum diese Sprache ihrem Ruf der Schnelligkeit und Freiheit in der Programmierung gerecht wird. Ein schnelles Antivirenprogramm und ein Utility in C, das für Ordnung in Ihrer Modulsammlung sorgt, runden das Informationsangebot mit praktischer Hilfestellung ab.

Natürlich fehlen auch in dieser neuesten Ausgabe die obligatorischen Utilities nicht – diesmal sind es mehr als ein halbes Dutzend. Darunter für den Komfortgewohnten eine Benutzeroberfläche für MS-DOS, eine Oberfläche mit Präprozessor für unseren SBasic-Compiler, ein Programm für »Sparsame«, nämlich ein sogenannter Packer, der Ihre Programme in einem Minimum an Speicherplatz unterbringt, und – mehr wird aber noch nicht verraten...

Ein Gedanke noch zum Schluß: Ein für uns überraschender, wenn auch sehr erfreulicher Aspekt kristallisierte sich mit der Leserumfrage der Ausgabe 6 heraus: Ein Gros der Leser bleibt der DOS Extra bereits seit der ersten Ausgabe, also seit fast eineinhalb Jahren, treu – wofür wir uns herzlich bedanken.

Ich hoffe, wir haben auch diesmal die richtige Mischung für Sie gefunden, und wünsche Ihnen eine informative Lektüre mit der DOS Extra 7.

Ihre

Eva Hierlmeier

Inhalt

Eingabehilfe:

SCheck 2.01:
Checksummer für sichere Listing-Eingabe 6
– *Eingabehilfe*

Grafik:

Die Programmierung der Hercules-Karte 10
– *Zugriff auf den Bildschirmspeicher*

Die Programmierung der CGA-Karte 13
– *Modi, Paletten, Aufbau des Bildspeichers*

Die Welt im Computer 20
– *Weltkarte aus Public-Domain-Koordinaten*

Zeichnen und leben lassen 23
– *Sprite-Editor*

PC intern:

Das Tor zur Zukunft 32
– *Steckplätze des PC und AT*

Der DMA-Controller 8237 36
– *Speicher schnell verschieben*

Der Interrupt-Controller 8259 38
– *Unterbrechungen unter Kontrolle*

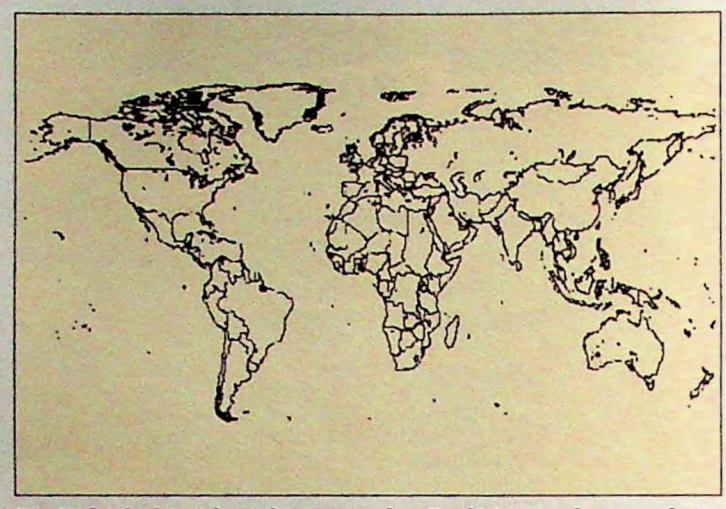
Der Zeitgeber 8253 40
– *Die Zeit im Griff*

Der Disketten-Controller 765 42
– *Zugriff auf den Massenspeicher*

Die parallele PC-Schnittstelle 45
– *Anschluß für den Drucker*

Die serielle PC-Schnittstelle 48
– *Informationen Bit für Bit*

Der Video-Controller 6845 50
– *Wie aus Bildern Daten werden*



Die Erde dreht sich nicht nur in der Realität, sondern auch im Computer – ein pfiffiges Grafiklisting eröffnet ungeahnte Möglichkeiten

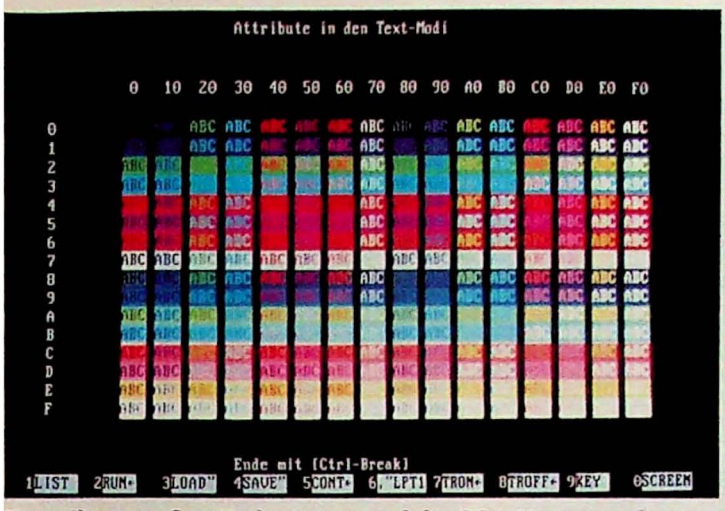
Seite 20

Ob Zeiger, Operatoren, Funktionen, Vektoren oder Präprozessor – der C-Schwerpunkt in dieser DOS Extra macht alle Themen transparent

Seite 54

In die mysteriösen Tiefen der Computer-Hardware bringt der ausführliche Schwerpunkt »PC intern« dieser DOS Extra endlich Licht

Seite 32



Trotz ihrer großen Verbreitung wird die CGA-Karte von den wenigsten Anwendern voll ausgenutzt – doch ab sofort ist ihre Programmierung kein Geheimnis mehr

Seite 13

Ein Präprozessor für Programme mit Marken statt Zeilennummern und eine Benutzeroberfläche machen SBasic komfortabel

Seite 92

Easydos, die Benutzeroberfläche zum Abtippen, wertet Ihr MS-DOS auf: nun ist es leistungsfähig und benutzerfreundlich

Seite 118

C:

Die Entwicklungsgeschichte von C **54**

Die Routine »printf« unter der Lupe **56**

Die Macht der Operatoren **58**

Zeiger sicher im Griff **60**

Flexibel mit Funktionen **63**

Präprozessor – der kleine Helfer **65**

Vektoren und Tabellen in C **67**

Dateizugriff in C **69**

Daten im Paket **72**
– *Der Datentyp »struct«*

Ordnung im Programmsalat **74**
– *Filterprogramme zur Modulverwaltung*

Virus im Visier **85**
– *Virenschutz für die Festplatte*

Utilities:

Benutzeroberfläche und Präprozessor für SBasic **92**

Maschinensprache inklusive **98**
– *Assemblerroutinen einbinden*

Residenten Programmen auf der Spur **102**

Bringen Sie Tempo in die Bildschirmausgabe **114**

Easydos: Damit Sie es leichter haben **118**
– *Benutzeroberfläche für MS-DOS*

Komprimieren Sie Ihre Programme **139**
– *Der DOS-Packer*

Kurven plotten ganz einfach **145**

Eine abstrakte Maschine **148**
– *Simulation eines Computersystems*

Tastatur und Treiber ohne Mystik **159**
– *Informationen über Tastaturcodes*

Tips & Tricks:

Tips & Tricks für MS-DOS **164**

Rubriken:

Editorial **3**

Impressum **170**

Inserenten **170**

SCheck 2.01: Checksummer für sichere Listing-Eingabe

Die Programme in der DOS International und der Sonderheftreihe DOS Extra werden nach einem speziellen Verfahren mit Prüfsummen versehen und abgedruckt. Das für die Überprüfung notwendige Prüfsummenprogramm »SCheck« liegt jetzt auch als Version 2.01 in Basic vor.

Alle in der DOS Extra und der DOS International abgedruckten Programme sind mit Prüfsummen und Zeilennummern versehen. Jede Programmzeile beginnt mit einer dreistelligen Prüfsumme. Sie ist in eckige Klammern gesetzt und wird beim Abtippen nicht mit eingegeben. Nach der Klammer folgt ein Leerzeichen, und darauf der eigentliche Text der Programmzeile (nur bei Basic-Programmen) oder eine Zeilennummer, die Sie ebenfalls nicht mit eingeben, und darauf erst der Programmtext.
Nur diesen letzten Teil der abgedruckten Zeile geben Sie beim Abtippen ein.
Bei allen Programmzeilen wird im Abdruck nach jeweils 55 Zeichen ein Zeilenumbruch vorgenommen. Die umgebrochene Teilzeile erhält dabei keine Prüfsumme. Bei der Eingabe geben Sie umgebrochene Zeilen in einem Stück ein, drücken also nicht die Enter-Taste wenn ein Zeilenumbruch im Ausdruck steht, sondern nur, wenn eine neue Zeilennummer auftaucht.
Beim Abtippen von Programmen sind Tippfehler kaum zu vermeiden. Mit dem Checksummenprogramm »SCheck« überprüfen Sie die abgetippten Listings auf eventuelle Fehler. Das Programm erlaubt nicht nur die Diagnose von fehlerhaft eingegebenen Zeilen, sondern mittels eines integrierten Editors auch gleich die Korrektur der fehlerhaften Zeilen.
SCheck wird unter GW-Basic mit

```
run "scheck"
```

gestartet. Als erstes wählen Sie die gewünschte Prüfsummenversion. In der vorliegenden DOS-Ausgabe brauchen Sie immer »SCheck Version 2.0«. Nur wer Programme aus Ausgaben der DOS International vor Oktober 1988 eingeben will, muß den ebenfalls integrierten Prüfsummenalgorithmus »SCheck Version 1.0« aufrufen. Danach geben Sie den gewünschten Prüfmodus – normalerweise »Standard« – an und abschließend den Programmnamen des Listings, so wie Sie es auf dem Datenträger abgelegt haben.
Zuvor sollten Sie sich aber sicherheitshalber eine Kopie Ihres Listings anfertigen. SCheck arbeitet zwar nicht mit dem Original, sondern mit einer temporären Zwischendatei und verändert die Originaldatei erst ganz zum Schluß, nach dem erfolgreichen Abschluß der Arbeit. Dennoch sollten Sie auch ein noch so geringes Risiko eines Datenverlusts, etwa durch Fehler beim Speichern, ausschalten.
Wie Sie im Bild sehen, zeigt SCheck die aktuelle Zeile stets in einem besonderen Bildschirmfenster an. Die Prüfsumme erscheint links unter dem Fenster. Beim Überprüfen Ihres Listings vergleichen Sie die angezeigte Prüfsumme mit der in der DOS-Ausgabe in eckigen Klammern abgedruckten Prüfsumme. Rechts daneben finden Sie die aktuelle Zeilennummer und den Dateinamen. Im unteren Fenster werden Fehlermeldungen und Hinweise auf die weitere Bearbeitung ausgegeben. Im abgebildeten Fall können Sie mit [Enter] die Zeile übernehmen oder mit [E] den Editor aufrufen.
Der Editor dient der Korrektur fehlerhafter Zeilen. Er arbeitet immer im Einfügemodus, so daß fehlerhafte Eingaben extra gelöscht werden müssen. Sie können also nichts versehentlich überschreiben. Mit den Cursor-Tasten steuern Sie den Cursor im Editierfenster. [Home] bringt den Cursor an den Anfang der angezeigten Zeile, [End] entsprechend ans Ende.
Löschen können Sie das Zeichen links vom Cursor mit der großen Löschtaste oberhalb der Enter-Taste, [Del] entfernt das Zeichen unter dem Cursor. Mit [Enter] übernehmen Sie die korrigierte Zeile. Steuerzeichen wie zum Beispiel Tabulatoren dürfen in einer Zeile nicht verwendet werden. Haben Sie so das gesamte Programm oder auch nur »verdächtige« Teile davon durchgearbeitet, so speichert ein letzter Druck auf die Enter-Taste das korrigierte Listing. SCheck wird beendet und das fehlerfreie Programmlisting steht zu Ihrer Verfügung.
SCheck erlaubt aber noch einen zweiten Weg, das Listing korrekt einzutippen. Dabei geben Sie die Prüfsumme gleich bei der Eingabe ein. Die eckigen Klammern lassen Sie aber weg. Hinter der dreistelligen Prüfsumme fügen Sie ein Leerzeichen ein und danach den normalen Programmtext dieser Zeile. Wenn Sie den Text so aufbereitet vorliegen haben, können Sie die automatische Überprüfung mit SCheck nutzen. Rufen Sie dazu die Option »Mit Prüfsumme« auf.
SCheck verwandelt das Listing nun in eine Quelldatei für den jeweiligen Compiler oder Interpreter, indem die drei Prüfsummenstellen und das Leerzeichen vom Rest der Zeile abge-

Von 2.0b zu 2.01

Als aufmerksamer Leser haben Sie sicher bereits bemerkt, daß die Nummer der hier vorgestellten SCheck-Version in Gegensatz zu der Version im letzten DOS Extra 2.01 lautet. Dieser »Update« ist nötig geworden, weil die Version 2.0b bei der Bearbeitung von sehr langen Programmzeilen einen »Overflow Error« erzeugt.
Falls Sie die Version 2.0b aus der letzten DOS Extra abgetippt haben, müssen Sie nur zwei kleine Änderungen vornehmen, um zur Version 2.01 zu gelangen. Ersetzen Sie dazu in Programmzeile 10070 den Ausdruck

```
DEFINT A-Z
```

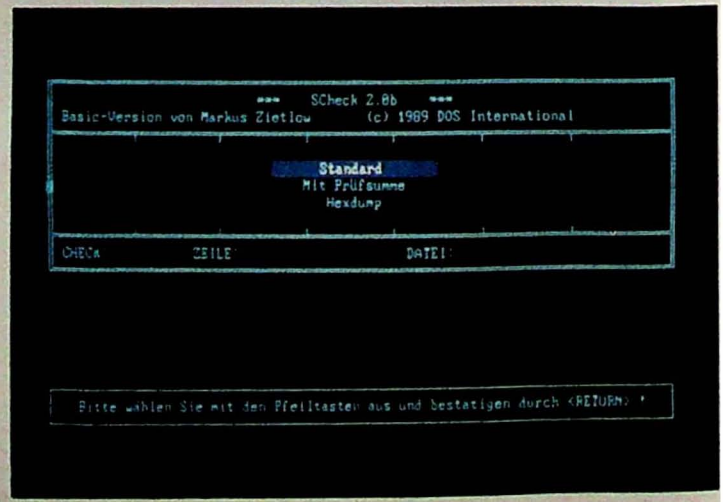
durch

```
DEFINT A-O,Q-Z:DEFSNG P
```

und fügen Sie die Zeile

```
12945 IF P>32767 THEN P = P - 32768
```

neu in das Programm ein.



Der Arbeitsbildschirm von SCheck 2.0b in Aktion

trennt werden. Aus der so entstandenen Programmzeile berechnet SCheck die Prüfsumme und vergleicht sie mit der zuvor abgetrennten Summe, die Sie eingegeben haben. Falls ein Unterschied auftritt, haben Sie Gelegenheit, die fehlerhafte Zeile manuell zu korrigieren.

Nach einem solchen Durchlauf von SCheck steht wieder das fehlerfreie Programm, diesmal allerdings ohne Prüfsumme, auf der Diskette. Auch hier gilt, daß Sie nicht mit dem Original arbeiten sollten, sondern nur mit einer Kopie, um im Zweifelsfall nochmal von vorne beginnen zu können.

SCheck verfügt aber noch über zwei weitere Optionen. Die von Fall zu Fall abgedruckten hexadezimalen Listings können Sie nämlich mit SCheck im Binärmodus direkt in lauffähige Programme umwandeln. Dazu müssen Sie das Hexdump-Listing mit einem ganz normalen Editor (zum Beispiel Turbo Pascal, Wordstar oder Edlin) als Textdatei eingeben. Anschließend überprüfen Sie mit SCheck zunächst die Korrektheit der abgetippten Daten wie bei jedem anderen Listing. Befindet sich das Programm korrekt auf Diskette, so wird die Option »Hexdump« aufgerufen. SCheck wird nun dazu veranlaßt, eine Hexdump-Textdatei in ein ablauffähiges Programm umzuwandeln. SCheck erzeugt in jedem Fall eine Datei mit der Namenserweiterung »bin«. Um daraus eine lauffähige Datei zu machen, müssen Sie sie mit dem Ren-Befehl von MS-DOS umbenennen:

```
ren <name>.bin <name>.com
```

Falls eine Exe-Datei erzeugt werden soll (bitte entnehmen Sie dies der jeweiligen Programmbeschreibung) lautet der entsprechende Befehl:

```
ren <name>.bin
<name>.exe
```

Sie sehen, mit SCheck 2.0b ist das korrekte Abtippen von Programmlistings kein bißchen komplizierter geworden – aber durch einen verbesserten Prüfsummen-Algorithmus um einiges sicherer.

(ev/ma)

SCheck mit SBasic compilieren

Um die Eingabehilfe SCheck mit dem SBasic-Compiler compilieren zu können, müssen Sie lediglich aus Programmzeile 10070 die Zeichenfolge »key off: « löschen und die Zeilen 10720, 11240 sowie 11110 bis 11170 vollständig entfernen, so daß damit die nicht unbedingt notwendige Fehlerbehandlung durch den On-Error-Befehl, den SBasic nicht verarbeiten kann, entfällt.

Name	SCheck.bas
Sprache	GW-Basic, SBasic
Grafikkarte	jede
Besonderheiten	keine

```
[SS5] 10000 '*****************************************
      *******
[F29] 10010 '* SCheck - Prüfsummengenerator für DOS-Lis
      tings *
[ML4] 10020 '*                                       *
[8Ø7] 10030 '*            Konzeption: Volker Everts  *
[AC7] 10040 '*          Basic-Version: Markus Zietlow *
[XS5] 10050 '*****************************************
      *******
[4U0] 10060 '
[FL7] 10070 KEY OFF: DEFINT A-O,Q-Z: DEFSNG P: COLOR 7,
      0,1
[5G8] 10080 T = 1: TEMP = 2: F = 3'Ein-/Ausgabekanäle f
      estlegen
[332] 10090 FALSE = 0
[902] 10100 TRUE = -1
[EY7] 10110 ZEICHEN$ = "0123456789ABCDEFGHKLMNPRSTUVWXY
      Z"
[9H2] 10120 LAUF = FALSE
[6B2] 10130 BIN = FALSE
[851] 10140 CLS
[9S1] 10150 PRINT
[K07] 10160 PRINT CHR$(201); STRING$(78, 205); CHR$(187
      );
[9FC] 10170 PRINT CHR$(186); STRING$(26, 32); "***    S
      Check 2.01    ***"; STRING$(27, 32); CHR$(186);
[Z7G] 10180 PRINT CHR$(186); " Basic-Version von Markus
       Zietlow       (c) 1989 DOS International
         "; CHR$(186);
[C5C] 10190 PRINT CHR$(204); "═══════════════════════
      ═══════════════════════════════════════════
      ═══"; CHR$(185);
[DR3] 10200 FOR ZAEHLER = 1 TO 5
[ZY6] 10210   PRINT CHR$(186); STRING$(78, 32); CHR$(18
      6);
[CR2] 10220 NEXT ZAEHLER
[84C] 10230 PRINT CHR$(204); "═══════════════════════
      ═══════════════════════════════════════════
      ═══"; CHR$(185);
[ØØD] 10240 PRINT CHR$(186); " CHECK:           ZEILE:
                        DATEI:
         "; CHR$(186);
[P07] 10250 PRINT CHR$(200); STRING$(78, 205); CHR$(188
      );
[2D2] 10260 LOCATE 20, 1
[B17] 10270 PRINT CHR$(218); STRING$(78, 196); CHR$(191
      );
[Ø07] 10280 PRINT CHR$(179); STRING$(78, 32); CHR$(179)
      ;
[C27] 10290 PRINT CHR$(192); STRING$(78, 196); CHR$(217
      );
[4V1] 10300 COLOR 7
[9F1] 10310 Y = 7
[9V6] 10320 LOCATE 7, 31: PRINT "SCheck  Version 1.x"
[PR6] 10330 LOCATE 8, 31: PRINT "SCheck  Version 2.0"
```

```
[SSG] 10340 LOCATE 21, 5: PRINT "Bitte wählen Sie mit d
      en Pfeiltasten aus und bestätigen durch <RETURN>
      !";
[2PD] 10350 COLOR 14, 1: FOR Q = 30 TO 50: LOCATE Y, Q:
       PRINT CHR$(SCREEN(Y, Q)): NEXT Q: COLOR 7, 0
[UP6] 10360 TASTE$ = INKEY$: IF TASTE$ = "" THEN 10360
[P96] 10370 IF TASTE$ <> CHR$(0) + CHR$(72) THEN 10400
[4ND] 10380 IF Y <> 7 THEN FOR Q = 30 TO 50: LOCATE Y,
      Q: PRINT CHR$(SCREEN(Y, Q)): NEXT Q: Y = Y - 1
[532] 10390 GOTO 10350
[1A6] 10400 IF TASTE$ <> CHR$(0) + CHR$(80) THEN 10430
[1ND] 10410 IF Y <> 8 THEN FOR Q = 30 TO 50: LOCATE Y,
      Q: PRINT CHR$(SCREEN(Y, Q)): NEXT Q: Y = Y + 1
[232] 10420 GOTO 10350
[F35] 10430 IF TASTE$ <> CHR$(13) THEN 10360
[8N2] 10440 FOR Q = 7 TO 8
[U16] 10450   LOCATE Q, 30: PRINT STRING$(21, 32)
[KN1] 10460 NEXT Q
[8Y2] 10470 VERSION = Y - 6
[GV1] 10480 COLOR 7
[NF1] 10490 Y = 7
[MA5] 10500 LOCATE 7, 36: PRINT "Standard"
[4A6] 10510 LOCATE 8, 34: PRINT "Mit Prüfsumme"
[C65] 10520 LOCATE 9, 37: PRINT "Hexdump"
[ZSG] 10530 LOCATE 21, 5: PRINT "Bitte wählen Sie mit d
      en Pfeiltasten aus und bestätigen durch <RETURN>
      !";
[9PD] 10540 COLOR 14, 1: FOR Q = 30 TO 50: LOCATE Y, Q:
       PRINT CHR$(SCREEN(Y, Q)): NEXT Q: COLOR 7, 0
[3R6] 10550 TASTE$ = INKEY$: IF TASTE$ = "" THEN 10550
[4B6] 10560 IF TASTE$ <> CHR$(0) + CHR$(72) THEN 10590
[BND] 10570 IF Y <> 7 THEN FOR Q = 30 TO 50: LOCATE Y,
      Q: PRINT CHR$(SCREEN(Y, Q)): NEXT Q: Y = Y - 1
[E32] 10580 GOTO 10540
[GA6] 10590 IF TASTE$ <> CHR$(0) + CHR$(80) THEN 10620
[CND] 10600 IF Y <> 9 THEN FOR Q = 30 TO 50: LOCATE Y,
      Q: PRINT CHR$(SCREEN(Y, Q)): NEXT Q: Y = Y + 1
[B32] 10610 GOTO 10540
[P35] 10620 IF TASTE$ <> CHR$(13) THEN 10550
[GN2] 10630 FOR Q = 7 TO 9
[126] 10640   LOCATE Q, 30: PRINT STRING$(21, 32)
[TN1] 10650 NEXT Q
[WG4] 10660 IF Y = 8 THEN LAUF = TRUE
[NA4] 10670 IF Y = 9 THEN BIN = TRUE
[RU5] 10680 LOCATE 12, 54: PRINT STRING$(25, 32)
[C9F] 10690 GOSUB 13150: LOCATE 21, 12: PRINT "Geben Si
      e bitte den Dateinamen der gewünschten Datei an !
      ";
[9P8] 10700 Y = 12: X = 54: LAENGE = 24: MENGE$ = "alle
      ": GOSUB 10890
[N6D] 10710 IF (TASTE$ = CHR$(27)) OR (Q$ = "") THEN LO
      CATE 12, 54: PRINT STRING$(25, 32): GOTO 10680
[LM3] 10720 ON ERROR GOTO 11160
[C74] 10730 OPEN Q$ FOR INPUT AS #T
[732] 10740 GOTO 11230
[302] 10750 COLOR 12
[2A6] 10760 LOCATE 8, 30: PRINT "D A T E I F E H L E R"
[VV1] 10770 COLOR 7
```

Listing. »SCheck.bas« für GW-Basic

```
[RØC] 1Ø78Ø GOSUB 1315Ø: LOCATE 21, 23: PRINT "Drücken
      Sie bitte irgendeine Taste !";
[B83] 1Ø79Ø WHILE INKEY$ = ""
[GX2] 1Ø8ØØ   PRINT CHR$(7);
[CE1] 1Ø81Ø WEND
[6U5] 1Ø82Ø LOCATE 8, 3Ø: PRINT STRING$(21, 32)
[842] 1Ø83Ø GOTO 1Ø68Ø
[2VØ] 1Ø84Ø '
[184] 1Ø85Ø '**********************************
[UD6] 1Ø86Ø '* Unterprogramm zur Dateneingabe *
[384] 1Ø87Ø '**********************************
[6VØ] 1Ø88Ø '
[GG1] 1Ø89Ø Q$ = ""
[EF2] 1Ø9ØØ LOCATE Y, X
[FS3] 1Ø91Ø PRINT Q$; CHR$(219);
[GR6] 1Ø92Ø TASTE$ = INKEY$: IF TASTE$ = "" THEN 1Ø92Ø
[ZK5] 1Ø93Ø IF (TASTE$ <> CHR$(8)) THEN 1Ø99Ø
[224] 1Ø94Ø   IF LEN(Q$) = Ø THEN 1Ø92Ø
[X44] 1Ø95Ø   Q$ = LEFT$(Q$, LEN(Q$) - 1)
[6N2] 1Ø96Ø   LOCATE Y, X
[V44] 1Ø97Ø   PRINT Q$; CHR$(219); " ";
[8A2] 1Ø98Ø   GOTO 1Ø92Ø
[T98] 1Ø99Ø IF TASTE$ = CHR$(13) THEN LOCATE Y, X: PRIN
      T Q$; " ";
[639] 11ØØØ IF (TASTE$ = CHR$(13)) OR (TASTE$ = CHR$(27
      )) THEN RETURN
[C55] 11Ø1Ø IF (ASC(TASTE$) < 32) THEN 1Ø92Ø
[NX4] 11Ø2Ø IF MENGE$ <> "zahl" THEN 11Ø5Ø
[ZC8] 11Ø3Ø   IF (ASC(TASTE$) < 48) OR (ASC(TASTE$) > 5
      7) THEN 1Ø92Ø
[892] 11Ø4Ø   GOTO 11Ø7Ø
[PU4] 11Ø5Ø IF MENGE$ = "alle" THEN 11Ø7Ø
[E86] 11Ø6Ø IF INSTR(MENGE$, TASTE$) = Ø THEN 1Ø92Ø
[YF6] 11Ø7Ø IF LEN(Q$) < LAENGE THEN Q$ = Q$ + TASTE$
[T22] 11Ø8Ø GOTO 1Ø9ØØ
[6H8] 11Ø9Ø Y = 1Ø: X = 65: LAENGE = 5: MENGE$ = "zahl"
      : GOSUB 1Ø89Ø
[BV5] 111ØØ IF (TASTE$ = CHR$(27)) THEN RETURN
[6UØ] 1111Ø '
[NK2] 1112Ø '******************
[TR3] 1113Ø '* Fehler-Routine *
[RK2] 1114Ø '******************
[AUØ] 1115Ø '
[KN1] 1116Ø CLOSE
[ZG2] 1117Ø RESUME 1Ø75Ø
[DUØ] 1118Ø '
[8G2] 1119Ø '******************
[TH3] 112ØØ '* Hauptprogramm *
[4G2] 1121Ø '******************
[BUØ] 1122Ø '
[KP3] 1123Ø COLOR 7: GOSUB 1315Ø
[N43] 1124Ø ON ERROR GOTO Ø
[K62] 1125Ø DATEI$ = Q$
[DF7] 1126Ø GROSS$ = DATEI$: GOSUB 13Ø4Ø: DATEI$ = GROS
      S$
[3W3] 1127Ø I = INSTR(DATEI$, ".")
[6K7] 1128Ø IF I = Ø THEN DATEI$ = DATEI$ + ".": I = LE
      N(DATEI$)
[U64] 1129Ø EXT$ = MID$(DATEI$, I, 4)
[8T4] 113ØØ DATEI$ = LEFT$(DATEI$, I - 1)
[HN5] 1131Ø LOCATE 12, 54: PRINT DATEI$; EXT$
[HB4] 1132Ø TEMPNAME$ = DATEI$ + ".$$$"
[5E6] 1133Ø OPEN TEMPNAME$ FOR OUTPUT AS #TEMP
[CZ7] 1134Ø IF BIN THEN OPEN DATEI$ + ".BIN" FOR OUTPUT
       AS #F
[CP7] 1135Ø IF VERSION = 2 THEN NUMMER = 1 ELSE NUMMER
      = Ø
[KYC] 1136Ø IF BIN THEN GOSUB 1315Ø: LOCATE 21, 25: PRI
      NT ">>> Binärdatei wird erzeugt <<<"
[TE3] 1137Ø WHILE NOT EOF(T)
[X24] 1138Ø   LINE INPUT #T, ZEILE$
[LDB] 11385   WHILE RIGHT$(ZEILE$,1)=" " : ZEILE$=LEFT$
      (ZEILE$,LEN(ZEILE$)-1) : WEND
[333] 1139Ø   WEITER = FALSE
[WT3] 1140Ø   IF WEITER THEN 115ØØ
[WN6] 1141Ø   GOSUB 1182Ø 'Gelesene Zeile anzeigen
[TS4] 1142Ø   IF BIN THEN WEITER = TRUE
[ZT3] 1143Ø   IF WEITER THEN 115ØØ
[AM6] 1144Ø   CH$ = INKEY$: IF CH$ = "" THEN GOTO 1144Ø
[3G3] 1145Ø   ERLAUBT = FALSE
[ZH8] 1146Ø   IF CH$ = CHR$(13) THEN ERLAUBT = TRUE: WE
      ITER = TRUE
[N1C] 1147Ø   IF (CH$ = "e") OR (CH$ = "E") THEN ERLAUB
      T = TRUE: GOSUB 12Ø6Ø'Zeile bearbeiten
[8ME] 1148Ø   IF (CH$ = "q") OR (CH$ = "Q") THEN IF NOT
       LAUF THEN GOSUB 1242Ø: IF SCHLUSS THEN GOTO 1163
      Ø'Quit
[F57] 1149Ø   IF ERLAUBT THEN GOTO 114ØØ ELSE GOTO 1144
      Ø
[8Y3] 115ØØ   PRINT #TEMP, ZEILE$
[DU3] 1151Ø   IF NOT BIN THEN 1161Ø
[D99] 1152Ø   WHILE LEFT$(ZEILE$, 1) = " ": ZEILE$ = MI
      D$(ZEILE$, 2): WEND
[364] 1153Ø   IF ZEILE$ = "" THEN 1161Ø
[K75] 1154Ø   B = VAL("&H" + LEFT$(ZEILE$, 2))
[TSH] 1155Ø   IF (B < Ø) OR (B > 255) THEN GOSUB 1315Ø:
       LOCATE 21, 22: PRINT "+++++ ABBRUCH WEGEN DATENF
      EHLER +++++": LOCATE 25, 1: END
[HN1] 1156Ø   BY = B
[TK3] 1157Ø   PRINT #F, CHR$(BY);
[YA4] 1158Ø   ZEILE$ = MID$(ZEILE$, 3)
[M99] 1159Ø   WHILE LEFT$(ZEILE$, 1) = " ": ZEILE$ = MI
      D$(ZEILE$, 2): WEND
[T92] 116ØØ   GOTO 1153Ø
[DY3] 1161Ø   NUMMER = NUMMER + 1
[8E1] 1162Ø WEND
[S7B] 1163Ø WHILE NOT EOF(T): LINE INPUT #T, ZEILE$: PR
      INT #TEMP, ZEILE$: WEND
[4P1] 1164Ø CLOSE
[8U5] 1165Ø OPEN DATEI$ + EXT$ FOR INPUT AS #1
[XØ6] 1166Ø OPEN DATEI$ + ".BAK" FOR OUTPUT AS #2
[XGB] 1167Ø WHILE NOT EOF(1): LINE INPUT #1, ZEILE$: PR
      INT #2, ZEILE$: WEND: CLOSE
[6D5] 1168Ø OPEN TEMPNAME$ FOR INPUT AS #1
[576] 1169Ø OPEN DATEI$ + EXT$ FOR OUTPUT AS #2
[UGB] 117ØØ WHILE NOT EOF(1): LINE INPUT #1, ZEILE$: PR
      INT #2, ZEILE$: WEND: CLOSE
[713] 1171Ø KILL TEMPNAME$
[GTN] 1172Ø GOSUB 1256Ø: LOCATE 12, 9: PRINT STRING$(11
      , 32): LOCATE 12, 25: PRINT STRING$(21, 32): LOCA
      TE 12, 53: PRINT STRING$(27, 32)'Statuszeile lösc
      hen
[494] 1173Ø IF BIN THEN EXT$ = ".BIN"
[N4H] 1174Ø GOSUB 1315Ø: LOCATE 21, (8Ø - (31 + LEN(DAT
      EI$ + EXT$))) / 2: PRINT "Die Datei '"; DATEI$ +
      EXT$; "' wurde gespeichert."
[CE2] 1175Ø LOCATE 25, 1
[H51] 1176Ø END
[4VØ] 1177Ø '
[TH3] 1178Ø '****************************
[B65] 1179Ø '* Gelesene Zeile anzeigen *
[NH3] 118ØØ '****************************
[2VØ] 1181Ø '
[BBB] 1182Ø IF LAUF THEN CHECK$ = "[" + LEFT$(ZEILE$, 3
      ) + "]": ZEILE$ = MID$(ZEILE$, 5)
[MDN] 1183Ø FOR Q = 1 TO LEN(CHECK$): IF ASC(MID$(CHECK
      $, Q, 1)) > 96 AND ASC(MID$(CHECK$, Q, 1)) < 123
      THEN MID$(CHECK$, Q, 1) = CHR$(ASC(MID$(CHECK$, Q
      , 1)) - 32)
[3P1] 1184Ø NEXT Q
[B57] 1185Ø LINKS = 2: OBEN = 6: RECHTS = 79: UNTEN = 1
      Ø
[CX4] 1186Ø LOCATE 12, 26: PRINT NUMMER
[Z45] 1187Ø GOSUB 1256Ø'Fenster löschen
[WX8] 1188Ø GOSUB 1266Ø'Zeile innerhalb des Fensters an
      zeigen
[366] 1189Ø GOSUB 128ØØ'Check-Summe errechnen
[ØM2] 119ØØ LOCATE 12, 1Ø
[HB2] 1191Ø PRINT PS$
[HM7] 1192Ø IF BIN THEN GOTO 12ØØØ'Binärdatei erzeugen
[N8C] 1193Ø IF LAUF THEN GOTO 1197Ø'Übereinstimmung der
       Prüfsummen feststellen
[X74] 1194Ø GOSUB 1315Ø: LOCATE 21, 4
[LKD] 1195Ø PRINT "[E] Editieren    [RETURN] Weiter
      [Q] Quit        [Ctrl-Break] Abbrechen"
[152] 1196Ø GOTO 1199Ø
[ATP] 1197Ø GOSUB 1315Ø: IF PS$ = CHECK$ THEN LOCATE 21
      , (8Ø - (22 + LEN(STR$(NUMMER)))) / 2: PRINT "Prü
      fsumme OK für Zeile"; NUMMER: WEITER = TRUE: GOTO
       12ØØØ
[1NH] 1198Ø LOCATE 21, 4: PRINT CHR$(7); "Prüfsummenfeh
      ler!  [E] Editieren  [RETURN] Weiter  [Ctrl-Break
      ] Abbrechen"
[XM2] 1199Ø LOCATE 12, 15
[KV1] 12ØØØ RETURN
[5UØ] 12Ø1Ø '
[MK2] 12Ø2Ø '******************
[4H3] 12Ø3Ø '* Zeile eingeben *
[PK2] 12Ø4Ø '******************
[9UØ] 12Ø5Ø '
[FVD] 12Ø6Ø GOSUB 1315Ø: LOCATE 21, 17: PRINT "Ändern S
      ie die Programmzeile !     [RETURN] Ende"
[245] 12Ø7Ø GOSUB 1256Ø'Fenster löschen
[7B5] 12Ø8Ø GOSUB 1266Ø'Zeile darstellen
[FL3] 12Ø9Ø LOCATE OBEN, LINKS
[7G3] 121ØØ X = 1: PRINT "■";
[8E2] 1211Ø CH$ = INKEY$
[FE6] 1212Ø IF LEFT$(CH$, 1) <> CHR$(Ø) THEN GOTO 1222Ø
[4N3] 1213Ø   CH$ = RIGHT$(CH$, 1)
[4NE] 1214Ø   IF CH$ = "M" THEN IF X > LEN(ZEILE$) THEN
       PRINT CHR$(7); : GOTO 1211Ø ELSE GOSUB 1234Ø: X
      = X + 1
[KZC] 1215Ø   IF CH$ = "K" THEN IF X = 1 THEN PRINT CHR
      $(7); : GOTO 1211Ø ELSE GOSUB 1234Ø: X = X - 1
[YZC] 1216Ø   IF CH$ = "H" THEN IF X < 79 THEN PRINT CH
      R$(7); : GOTO 1211Ø ELSE GOSUB 1234Ø: X = X - 78
[YCF] 1217Ø   IF CH$ = "P" THEN IF X + 77 > LEN(ZEILE$)
       THEN PRINT CHR$(7); : GOTO 1211Ø ELSE GOSUB 1234
      Ø: X = X + 78
```

```
[TV7] 1218Ø   IF CH$ = "G" THEN IF X <> 1 THEN GOSUB 12
      34Ø: X = 1
[41C] 1219Ø   IF CH$ = "O" THEN IF X <> LEN(ZEILE$) + 1
       THEN GOSUB 1234Ø: X = LEN(ZEILE$) + 1
[8PM] 122ØØ   IF CH$ = "S" THEN IF X <= LEN(ZEILE$) THE
      N ZEILE$ = LEFT$(ZEILE$, X - 1) + MID$(ZEILE$, X
      + 1): GOSUB 1266Ø ELSE PRINT CHR$(7); : GOTO 1211
      Ø
[G92] 1221Ø   GOTO 1224Ø
[XXN] 1222Ø IF CH$ = CHR$(8) THEN IF X > 1 THEN ZEILE$
      = LEFT$(ZEILE$, X - 2) + MID$(ZEILE$, X): GOSUB 1
      266Ø: GOSUB 1234Ø: X = X - 1 ELSE PRINT CHR$(7);
      : GOTO 1211Ø
[LZN] 1223Ø IF CH$ >= " " THEN IF LEN(ZEILE$) = 255 THE
      N PRINT CHR$(7); : GOTO 1211Ø ELSE ZEILE$ = LEFT$
      (ZEILE$, X - 1) + CH$ + MID$(ZEILE$, X): GOSUB 12
      66Ø: X = X + 1
[7YB] 1224Ø LOCATE OBEN + INT((X - 1) / 78), LINKS - 1
      + X - INT((X - 1) / 78) * 78: PRINT "■":
[LK5] 1225Ø IF CH$ <> CHR$(13) THEN GOTO 1211Ø
[648] 1226Ø IF LAUF THEN ZEILE$ = MID$(CHECK$, 2, 3) +
      " " + ZEILE$
[XL6] 1227Ø GOSUB 1234Ø: GOSUB 1266Ø'Zeile anzeigen
[2W1] 1228Ø RETURN
[NUØ] 1229Ø '
[CP3] 123ØØ '****************************
[2H5] 1231Ø '* Zeichen wiederherstellen *
[EP3] 1232Ø '****************************
[LUØ] 1233Ø '
[UM9] 1234Ø LOCATE OBEN + INT((X - 1) / 78), LINKS - 1
      + X - INT((X - 1) / 78) * 78
[YXA] 1235Ø IF MID$(ZEILE$, X, 1) = "" THEN PRINT " " E
      LSE PRINT MID$(ZEILE$, X, 1)
[4W1] 1236Ø RETURN
[RUØ] 1237Ø '
[ØA2] 1238Ø '***************
[3X2] 1239Ø '* Quit (Ende) *
[W92] 124ØØ '***************
[NUØ] 1241Ø '
[U63] 1242Ø SCHLUSS = FALSE
[KYB] 1243Ø SICHER$ = "": FOR Q = 2 TO 79: SICHER$ = SI
      CHER$ + CHR$(SCREEN(21, Q)): NEXT Q
[7LD] 1244Ø GOSUB 1315Ø: LOCATE 21, 16: PRINT "Drücken
      Sie bitte [J], um die Arbeit zu beenden !"
[GA6] 1245Ø CH$ = INKEY$: IF CH$ = "" THEN GOTO 1245Ø
[R3B] 1246Ø IF (CH$ = "j") OR (CH$ = "J") THEN SCHLUSS
      = TRUE: PRINT #TEMP, ZEILE$
[BA2] 1247Ø GOSUB 1315Ø
[P78] 1248Ø IF NOT SCHLUSS THEN LOCATE 21, 2: PRINT SIC
      HER$
[BW1] 1249Ø RETURN
[651] 125ØØ END
[TUØ] 1251Ø '
[6R2] 1252Ø '*******************
[U24] 1253Ø '* Fenster löschen *
[8R2] 1254Ø '*******************
[XUØ] 1255Ø '
[Ø44] 1256Ø FOR Q = OBEN TO UNTEN
[Z53] 1257Ø LOCATE Q, LINKS
[8B6] 1258Ø   PRINT STRING$(RECHTS - LINKS + 1, 32)
[ZN1] 1259Ø NEXT Q
[AW1] 126ØØ RETURN
[XUØ] 1261Ø '
[265] 1262Ø '*****************************************
      *
[7W7] 1263Ø '* Inhalt von "zeile$" im Fenster anzeigen
      *
[465] 1264Ø '*****************************************
      *
[1VØ] 1265Ø '
```

```
[SR2] 1266Ø F.X = Ø: F.Y = Ø
[CC5] 1267Ø FOR ZAEHLER = 1 TO LEN(ZEILE$)
[F85] 1268Ø   LOCATE OBEN + F.Y, LINKS + F.X
[UH5] 1269Ø   PRINT MID$(ZEILE$, ZAEHLER, 1);
[SK2] 127ØØ   F.X = F.X + 1
[597] 1271Ø   IF F.X >= RECHTS - 1 THEN F.X = Ø: F.Y =
      F.Y + 1
[6S2] 1272Ø NEXT ZAEHLER
[RX6] 1273Ø LOCATE OBEN + F.Y, LINKS + F.X: PRINT " ";
[KW1] 1274Ø RETURN
[5VØ] 1275Ø '
[AH4] 1276Ø '************************************
[ET6] 1277Ø '* Zeile auf Check-Summe überprüfen *
[CH4] 1278Ø '************************************
[9VØ] 1279Ø '
[F42] 128ØØ P = Ø: A = 1
[8SB] 1281Ø WHILE (MID$(ZEILE$, A, 1) = " ") OR (MID$(Z
      EILE$, A, 1) = CHR$(9)): A = A + 1: WEND
[374] 1282Ø FOR I = A TO LEN(ZEILE$)
[4CX] 1283Ø   IF MID$(ZEILE$, I, 1) >= " " THEN GROSS$
      = MID$(ZEILE$, I, 1): IF VERSION = 1 THEN P = ((I
       - A + 1) AND 7) * ASC(GROSS$) + P: GOTO 1292Ø EL
      SE GOSUB 13Ø4Ø: P = (((I - A + 1) AND 3) + 1) * A
      SC(GROSS$) + P: GOTO 1292Ø
[S22] 1284Ø   COLOR 12
[5N6] 1285Ø   LOCATE 8, 3Ø: PRINT "D A T E I F E H L E
      R"
[Ø22] 1286Ø   COLOR 7
[K2C] 1287Ø   LOCATE 21, 13: PRINT "          Drücken S
      ie bitte irgendeine Taste !          ";
[CC3] 1288Ø   WHILE INKEY$ = ""
[D93] 1289Ø     PRINT CHR$(7);
[5L1] 129ØØ   WEND
[R16] 1291Ø   LOCATE 8, 3Ø: PRINT STRING$(21, 32)
[ØP1] 1292Ø NEXT I
[KF2] 1293Ø PS$ = "[    ]"
[AM2] 1294Ø FOR I = 2 TO 4
[K25] 12945   IF P > 32767 THEN P = P - 32768
[5T7] 1295Ø   MID$(PS$, I, 1) = MID$(ZEICHEN$, (P AND 3
      1) + 1, 1)
[HZ2] 1296Ø   P = FIX(P / 32)
[5P1] 1297Ø NEXT I
[YW1] 1298Ø RETURN
[HVØ] 1299Ø '
[ZX3] 13ØØØ '********************************
[KØ6] 13Ø1Ø '* In Großbuchstaben umwandeln *
[934] 13Ø2Ø '********************************
[AUØ] 13Ø3Ø '
[9ZN] 13Ø4Ø FOR Q = 1 TO LEN(GROSS$): IF ASC(MID$(GROSS
      $, Q, 1)) > 96 AND ASC(MID$(GROSS$, Q, 1)) < 123
      THEN MID$(GROSS$, Q, 1) = CHR$(ASC(MID$(GROSS$, Q
      , 1)) - 32)
[8T8] 13Ø5Ø IF MID$(GROSS$, Q, 1) = "ä" THEN MID$(GROSS
      $, Q, 1) = "Ä"
[NU8] 13Ø6Ø IF MID$(GROSS$, Q, 1) = "ö" THEN MID$(GROSS
      $, Q, 1) = "Ö"
[UU8] 13Ø7Ø IF MID$(GROSS$, Q, 1) = "ü" THEN MID$(GROSS
      $, Q, 1) = "Ü"
[DN1] 13Ø8Ø NEXT Q
[YV1] 13Ø9Ø RETURN
[BUØ] 131ØØ '
[YZ2] 1311Ø '**********************
[9N4] 1312Ø '* Meldezeile löschen *
[ØØ3] 1313Ø '**********************
[FUØ] 1314Ø '
[ZT5] 1315Ø LOCATE 21, 2: PRINT STRING$(78, 32)
[ZV1] 1316Ø RETURN
```

Listing. »SCheck.bas« für GW-Basic

Utilities gesucht

Programme, die den Bildschirminhalt auf den Kopf stellen, suchen unsere Leser sicherlich nicht. Aber einen wissenschaftlichen Taschenrechner, ein Festplatten-Backup-Programm, einen Extended-Memory-Treiber, Druckerspooler oder Cache-Speicher. Zögern Sie deshalb nicht, schicken Sie uns unverbindlich die aktuelle Version Ihres Programms und einen Ausdruck der Dokumentation.
Falls Sie sich vorweg informieren wollen, zwischen 10.00 und 12.00 Uhr und zwischen 15.00 und 17.00 Uhr stehen wir Ihnen an allen Werktagen unter der Rufnummer (089) 4 61 00 10 zur Verfügung.

Schicken Sie Ihre Unterlagen an folgende Adresse:
Redaktionsbüro Everts & Hagedorn
»DOS Software«
Hans-Stießberger-Str. 3
8013 Haar/München

Die Programmierung der Hercules-Karte

Der Benutzer einer Hercules-Karte hat besondere Probleme, wenn er die Grafikkarte programmieren will, weil nicht alle Merkmale dieser Karte von MS-DOS beziehungsweise vom BIOS unterstützt werden. Da die Dokumentation häufig noch sehr knapp gehalten ist, wird die Programmierung der Hercules-Karte schnell zum Ratespiel.

Die Hercules-Karte ist trotz ihres Alters immer noch sehr beliebt, weil diese Karte sowohl ein optisch ansprechendes Schriftbild als auch hochauflösende Grafik bietet. Ein Manko besteht darin, daß sich bei dieser Karte nur zwei Farben (beziehungsweise Schwarz und eine Farbe) darstellen lassen.

Die Hercules-Karte kann allerdings in zwei Modi betrieben werden. Es existiert ein Textmodus, in dem keine Grafik dargestellt werden kann, und es gibt einen Grafikmodus, der sowohl Text wie auch Grafik auf den Bildschirm bringt.

Jedoch gestaltet sich die Darstellung von Text im letzteren Modus nicht so einfach wie bei anderen Grafikkarten, da das BIOS keine Unterstützung für den Grafikmodus enthält.

Die Hercules-Karte besteht, wie andere Grafikkarten auch, aus zwei Komponenten: zum einen aus der Steuerlogik, die das Bild auf dem Bildschirm ausgibt, und zum anderen aus dem Bildschirmspeicher, in dem das Bild in digitaler Form für die Steuerlogik bereitgehalten wird.

Der Bildschirmspeicher wird auf zwei Arten verwendet: – Im Textmodus werden ASCII-Codes in den Bildschirmspeicher geschrieben. Die Grafikkarte wandelt diese Codes in Punktmuster um, die wiederum auf dem Bildschirm gezeigt werden.

– Im Grafikmodus werden einzelne Punkte in Form verschlüsselter Farben (Schwarz oder Weiß) in den Speicher geschrieben. Die Punkte werden ohne weitere Umwandlung lediglich zum Bildschirm geschickt.

Der Bildschirmspeicher ist ein reservierter Bereich im Gesamtspeicher, der sowohl vom Prozessor (das heißt vom Programm) als auch von der Bildschirmsteuerlogik gelesen und beschrieben werden kann. Dieser Speicher ist physikalisch getrennt vom »normalen« Arbeitsspeicher, in dem Programme und Daten abgelegt werden. Diese Trennung spiegelt sich auch in den Adressen wider: Während der Arbeitsspeicher eines PC von der hexadezimalen Adresse »0000:0000« bis maximal »9000:FFFF« liegt, ist der Bildschirmspeicher der Hercules-Karte standardmäßig zwischen den Adressen »B000:0000« und »B000:7FFF« angesiedelt und damit 32 KByte groß. In der letzten Zeit ist es aber üblich geworden, daß eine Hercules-Karte 64 KByte Bildschirmspeicher erhält. In diesem Fall liegt der Speicher zwischen den Adressen »B000:0000« und »B000:FFFF«.

Der Basisaufbau des Bildschirmspeichers im Textmodus ist in Bild 1 zu sehen. Wie abgebildet, besteht jedes angezeigte Zeichen aus 2 Byte. Das erste Byte ist einfach das ASCII-Zeichen, während das zweite Byte das sogenannte »Attribut« darstellt. Die verfügbaren Attribute sind in der Tabelle aufgeführt. Allerdings verwirklichen nicht alle herculeskompatiblen Karten alle Attribute, ferner sind nicht alle Monitore in der Lage, die verschiedenen Helligkeitsstufen abzubilden. Sie müssen daher prüfen, welche Attribute von Ihrer Grafikkarte und Ihrem Bildschirm dargestellt werden. Zu diesem Zweck führt das Basic-Programm in Listing 1 die verschiedenen Attribut-Bits vor.

Der Aufbau des Bildschirmspeichers ist im Textmodus sehr einfach: An der Adresse »B000:0000« steht das erste ASCII-Zeichen, das in der oberen, linken Ecke des Bildschirms gezeigt wird. In der nächsten Speicherzelle, Adresse »B000:0001«, steht das Attribut-Byte für dieses Zeichen. Eine Zelle weiter, in »B000:0002«, befindet sich schon das zweite ASCII-Zeichen, danach das dazugehörende Attribut und so weiter. Da jedes Zeichen aus 2 Byte besteht, gibt folgende Formel die Verschiebungen eines Zeichens relativ zum ersten Zeichen an:

```
(Zeile x 160) + (Spalte x 2)
```

Die Adresse des zugehörigen Attribut-Bytes ist immer die Adresse des Zeichens plus eins.

Eine schnelle Berechnung zeigt, daß die Hercules-Karte über viel mehr Speicher verfügt, als im Textmodus benutzt wird. Für 80 Spalten und 25 Zeilen werden 4000 Byte benötigt (80x25x2) – die Karte besitzt aber 32 beziehungsweise 64 KByte. Im Gegensatz zu anderen Karten bietet die Hercules-Karte standardmäßig keine Methode, über die der restliche Speicher genutzt werden könnte. Manche kompatiblen Karten bieten jedoch eine zweite Textseite; dies ist wiederum ein Fall, in dem Sie prüfen müssen, ob Ihre Karte dieses Merkmal aufweist. Um zur zweiten Textseite zu schalten, verwenden Sie die beiden Basic-Anweisungen:

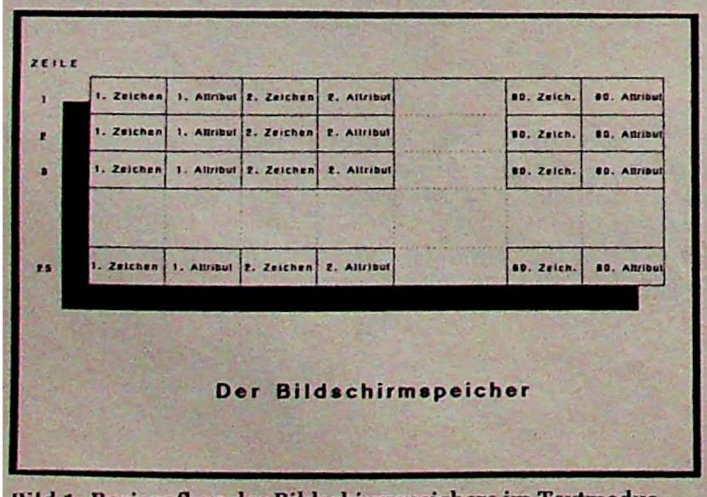


Bild 1. Basisaufbau des Bildschirmspeichers im Textmodus

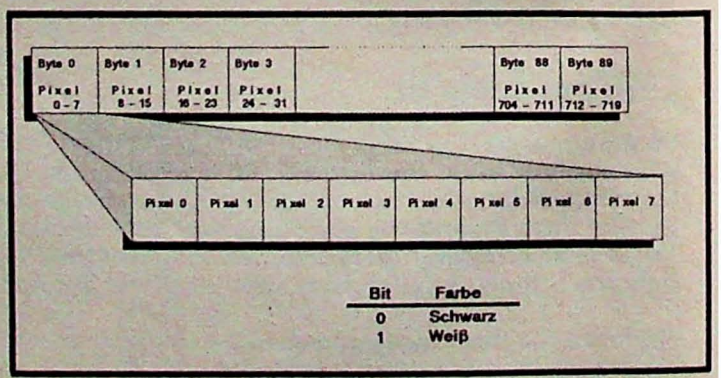


Bild 2. Logischer Aufbau des Bildschirmspeichers bei der Hercules-Karte

```
out &H3BF,3
out &H3B8,inp(&H3B8) or &H80
```

Nach diesen Anweisungen wird der Inhalt der zweiten Hälfte des Bildschirmspeichers (ab der Adresse »B000:8000«) gezeigt. Weil die zweite Seite nicht vom BIOS unterstützt wird, müssen Sie eine eigene Routine benutzen, um Zeichen in diese Seite zu schreiben. Wenn Sie zurück zur ersten Textseite schalten wollen, genügt folgende Anweisung

```
out &3B8,inp(&H3B8) and &H7F
```

Diese Umschaltung funktioniert, wie gesagt, nur bei manchen kompatiblen Karten; wenn nichts auf dem Bildschirm erscheint, obwohl Sie korrekt zur zweiten Seite umgeschaltet und geschrieben haben, dann bietet Ihre Karte eben keine zweite Seite.

Wenn Sie ein Zeichen direkt in den Bildschirmspeicher schreiben, sehen Sie dieses Zeichen unmittelbar auf dem Bildschirm.

Voraussetzung für das direkte Schreiben auf den Bildschirm ist, daß die Adresse des Zeichens berechnet wird. Zu diesem Zweck benutzen Sie die Formel von oben. Das Schreiben läuft dann ganz einfach ab: Die Adresse des Zeichens wird berechnet, anschließend das Zeichen im ASCII-Code an diese Adresse geschrieben und das Attribut in der nachfolgenden Adresse abgelegt. Dieser Ablauf ist in der Basic-Routine in Listing 2 verwirklicht.

Eine ebenso nützliche Funktion ist das Lesen eines Zeichens von einer bestimmten Bildschirmposition. Obwohl GW-Basic dies in seinem Screen-Befehl anbietet, geht es wie bei der Ausgabe auch hier einfacher und schneller mit direkten Zugriffen auf den Bildschirmspeicher. Außerdem bekommen Sie das Attribut eines Zeichens nur mit einem direkten Zugriff.

Wert	Art der Anzeige
0	Unsichtbar (schwarzes Zeichen auf schwarzem Hintergrund)
1	Unterstrichen
7	Normal
9	Hell und unterstrichen
F	Hell
70	Invers (schwarzes Zeichen auf weißem Hintergrund)
78	Dunkles Zeichen auf normalem (weißem) Hintergrund
81	Unterstrichen und blinkend
87	Blinkend
88	Unsichtbar (dunkles Zeichen auf dunklem Hintergrund)
89	Hell, unterstrichen und blinkend
8F	Hell und blinkend
F0	Invers und blinkend
F8	Dunkles Zeichen auf hellem Hintergrund

Tabelle: Die Attribute der Hercules-Karte im Textmodus

In Listing 3 ist die Basic-Routine zu finden, die das gelesene ASCII-Zeichen in der Variablen »zcode« und das gelesene Attribut in der Variablen »zattr« zurückgibt. Bei beiden Routinen müssen Sie lediglich den Wert 8000hex zu der berechneten Adresse addieren, wenn die Aus- beziehungsweise Eingabe die zweite Textseite ansprechen soll.

Die nächsten Funktionen ändern den Cursor. Die erste setzt die Position des Cursors, die zweite fragt die Position des Cursors ab. Der Locate-Befehl setzt die Cursor-Position, und »Csrlin« und »Pos« geben die aktuelle Cursor-Position zurück.

Die letzte in diesem Zusammenhang vorzustellende Routine bestimmt die Größe des Cursors. Die Punktmatrix eines Zeichens besteht bei der Hercules-Karte aus vierzehn Reihen zu je neun Punkten. Die Reihen werden von oben (Reihe 0) nach unten (Reihe 13) numeriert. Mit der BIOS-Routine »Cursorgröße setzen« legen Sie fest, aus welchen Reihen der Cursor besteht. Diese Routine ist in Basic bereits als Teil des Locate-Befehls vorgesehen. Zum Beispiel erzeugt der Befehl

```
locate ,,,13,13
```

den bekannten Unterstrich als Cursor, während der Befehl

```
locate ,,,0,13
```

einen Block als Cursor ausgibt. Auch in Basic ist der Cursor problemlos auszuschalten (unsichtbar):

```
locate ,,0
```

Ein weiterer Tip: Es gibt keine Funktion, welche die aktuelle Größe des Cursor zurückgibt. Hierzu bietet jedoch das BIOS Hilfe an: In den beiden Bytes der Adresse 460hex stehen die Anfangs- und Endreihen des Cursors. Seltsamerweise steht das Ende vor dem Anfang. Diese beiden Bytes können Sie mit folgenden Basic-Zeilen lesen:

```
def seg=&H40
Ende=peek(&H60)
Anfang=peek(&H61)
```

Das erste Problem beim Grafikmodus der Hercules-Karte: Wie wird dieser Modus eingeschaltet? Weder im BIOS noch in Basic ist dieser Modus vorgesehen. Deshalb kann in diesem Modus nur mit eigenen Routinen gearbeitet werden. Die Routine in Listing 4 schaltet den Monitor aus, um ihn vor gefährlichen Spannungen während des Wechsels zu schützen. In der Schleife wird die Steuerlogik auf der Hercules-Karte neu programmiert und anschließend der Monitor wieder eingeschaltet, aber der Inhalt des Bildschirmspeichers nicht gelöscht.

Wenn Sie zurück in den Textmodus schalten wollen, benötigen Sie eine ähnliche Routine, um die Grafikkarte neu zu programmieren. Die Basic-Anweisung »Screen« reicht dazu nämlich nicht aus. Eine Basic-Routine, welche die Hercules-Karte für Textmodus programmiert, sehen Sie in Listing 5.

Wie bereits erwähnt, gestaltet sich die Arbeitsweise der Hercules-Karte im Grafikmodus wesentlich anders als im Textmodus. Dieser Unterschied wird schon aus dem logischen Aufbau des Bildschirmspeichers ersichtlich (Bild 2). Zunächst sieht der Speicher ganz »ordentlich« aus; die ersten 90 Byte enthalten die erste Grafikpunktzeile (Zeile Nummer 0). Die zweite Grafikzeile (Zeile Nummer 1) steht jedoch nicht – wie beim Textmodus – direkt nach der ersten, sondern im zweiten Viertel des Bildschirmspeichers; genauer – sie steht an der Adresse »B000:2000«. Die dritte Zeile im Grafikmodus beginnt bei Adresse »B000:4000« und die vierte Zeile bei Adresse »B000:6000«. Die fünfte Zeile steht direkt nach der ersten, die sechste folgt nach der zweiten und so weiter. Der Bildschirmspeicher im Grafikmodus ist demzufolge in vier Abschnitte geteilt. Jeder Abschnitt beinhaltet ein Viertel der Zeilen: Der erste enthält also die Zeilen mit den Nummern 0, 4, 8 und so weiter; der zweite die Zeilen 1, 5, 9 und so weiter; im dritten Abschnitt befinden sich die Zeilen 2, 6, 10 etc. und im vierten die Zeilen 3, 7, 11 etc.

Wie bei anderen älteren Grafikkarten auch, ergibt sich diese verwirrende Zusammenstellung der Grafikzeilen im Speicher aus der Tatsache, daß die Hardware dadurch ein bißchen einfacher wurde.

Eine Formel für die Anfangsadresse einer Grafikzeile entwickelt sich demgemäß:

```
(Zeile mod 4)x8192+(int (Zeile\4)x90)
```

Diese Informationen versetzen Sie bereits in die Lage, ein paar Programmzeilen zu erfinden, die einen direkten Zugriff auf den Bildschirmspeicher im Grafikmodus zur Folge haben.

Es gibt für die Hercules-Karte keine BIOS-Routinen, um Bildpunkte zu schreiben und zu lesen, auch die Basic-Funktionen »pset«, »preset« und »point« funktionieren nicht mit einer Herkules-Karte. Sie müssen deswegen eigene Routinen erfinden, um diesen Modus auszunutzen. Die entsprechende Basic-Routine zum Schreiben eines Punktes finden Sie in Listing 6.

Die Berechnung der Adresse ist schon bekannt, aber auch die Bitposition innerhalb des Bytes an dieser Adresse muß berechnet werden. Wie Sie bereits in Bild 2 gesehen haben, enthält jedes Byte im Bildschirmspeicher 8 Bildpunkte. Der linke Punkt befindet sich im obersten Bit, der

befindet sich im obersten Bit, der rechte Punkt im untersten. Eine sehr einfache und schnelle Methode zur »Berechnung« der Bit-Position besteht darin, alle acht »Masken« in Variablen zu speichern. Im Programm wird dann die Spalte benutzt, um das entsprechende Bit auszuwählen:

```
dim masken%(8)
masken%(0)=&h80: masken%(1)=&h40
masken%(2)=&h20: masken%(3)=&h10
masken%(4)=&h08: masken%(5)=&h04
masken%(6)=&h02: masken%(7)=&h01
maske = masken% (spalte and 7)
```

Dies ist auch die Methode, die in der Routine in Listing 6 Verwendung findet. In der Zeile 10040 wird das Byte aus dem Bildschirmspeicher gelesen, um die anderen sieben Punkte wieder unverändert schreiben zu können. Mit »Maske« löschen Sie das Bit, das zu dem zu schreibenden Punkt gehört, und danach setzen Sie das Bit, um die gewünschte »Farbe« zu erhalten. Das Ganze wird sofort wieder in den Bildschirmspeicher geschrieben, und die Routine kehrt zurück.

Wenn Sie einen Punkt lesen wollen, gehen Sie ähnlich vor. Die Adresse und die Maske werden gleich berechnet, aber nicht die gesuchte Farbe. Wenn Sie die Farbe herausgelesen haben, müssen Sie lediglich das Bit ganz nach rechts bringen, um eine »0« oder eine »1« zurückzuliefern. Den Vorgang können Sie nach dem Beispiel in Listing 6 programmieren; in der neuen Routine fällt lediglich die Verschiebung des Bit anders aus als in Listing 6, die Formeln für die Adresse und die Maske bleiben gleich.

Im Grafikmodus benötigt die Hercules-Karte (720x348)/8 Byte oder etwa 30,6 KByte. So kann eine Hercules-Karte mit 32 KByte nur eine Grafikseite darstellen. Die Karten mit 64 KByte bieten aber zwei Grafikseiten an. Selbstverständlich ist zu einem Zeitpunkt nur eine Seite sichtbar, jedoch können Sie Ihre Programme so schreiben, daß ein Bild in der unsichtbaren Seite aufgebaut und anschließend durch Seitenwechsel in seiner Gesamtheit auf den Bildschirm gebracht wird. Diese Vorgehensweise ist für den Betrachter optisch angenehmer, als wenn sich das Bild Punkt für Punkt aufbauen würde.

Der Seitenwechsel im Grafikmodus ist ähnlich dem im Textmodus. Sie benötigen genau eine Basic-Anweisung, um die Seite zu wählen:

```
out &h3B8, inp(&h3b8) or &h80
```

wählt die zweite Grafikseite, während

```
out &h3B8, inp(&h3b8) and &h7f
```

die erste Seite bestimmt. Wie im Textmodus müssen Sie lediglich 8000hex zu den Adressen von Listing 5 addieren, um diese Routinen auf der zweiten Grafikseite umzustellen.

(Thomas Little/al)

```
[918] 1Ø ' alle Attribute des Text-Modus werden gezeigt
[8UØ] 2Ø CLS
[6B8] 3Ø LOCATE 1,23: PRINT "Attribute der Herkules-Kar
    te";
[GZ1] 4Ø FOR I=Ø TO 15
[2M4] 5Ø   LOCATE I+5,3: PRINT HEX$(I)
[8D5] 6Ø   LOCATE 3,I*4+9: PRINT HEX$(I*16);
[W61] 7Ø   NEXT
[632] 8Ø FOR I=Ø TO 255
[AU4] 9Ø   ADR=(I MOD 16)*16Ø+(I\16)*8+656
[DL2] 1ØØ   DEF SEG=&HBØØØ
[RK4] 11Ø   POKE ADR,&H58: POKE ADR+1,I
[1D1] 12Ø   NEXT
[X88] 13Ø LOCATE 22,23: PRINT "Ende mit irgendeiner Tas
    te";
[LC3] 14Ø IF INKEY$="" THEN 14Ø
[R41] 15Ø CLS
[831] 16Ø END
```

Listing 1. »h-attr.bas« führt die Attribut-Bits der Hercules-Karte vor

```
[PT2] 9999  ' Subroutine
[B38] 1ØØØØ IF (Z<1) OR (SP<1) OR (SP>8Ø) OR (Z>25) THE
    N RETURN
[3Y2] 1ØØ1Ø DEF SEG = &HBØØØ
[S97] 1ØØ2Ø MM=(Z-1)*16Ø+(SP-1)*2: POKE MM,C: POKE MM+1
    ,A
[5S1] 1ØØ3Ø DEF SEG
[GV1] 1ØØ4Ø RETURN
```

Listing 2. Das Schreiben eines Zeichens bei Einsatz der Hercules-Karte gelingt mit »h-write.bas«

```
[BL2] 9999 ' Subroutine
[YZ1] 1ØØØØ C=-1: A=-1
[C38] 1ØØ1Ø IF (Z<1) OR (SP<1) OR (SP>8Ø) OR (Z>25) THE
    N RETURN
[4Y2] 1ØØ2Ø DEF SEG = &HBØØØ
[XB7] 1ØØ3Ø MM=(Z-1)*16Ø+(SP-1)*2: C=PEEK(MM): A=PEEK(M
    M+1)
[6S1] 1ØØ4Ø DEF SEG
[HV1] 1ØØ5Ø RETURN
```

Listing 3. Die Basic-Routine »h-lesen.bas«, die das gelesene ASCII-Zeichen in der Variablen »ZCode« und das gelesene Attribut in der Variablen »ZAttr« zurückgibt

```
[177] 999  ' Diese Routine stellt Grafik-Modus ein
[F56] 1ØØØ OUT &H3BF,1  ' Grafik-Modus erlauben
[FU5] 1Ø1Ø OUT &H3B8,Ø  ' Monitor abschalten
[ZP7] 1Ø2Ø ' Die nächsten 9 Zeilen setzen Grafik-Modus
[6B4] 1Ø3Ø OUT &H3B4,Ø: OUT &H3B5,&H35
[PB4] 1Ø4Ø OUT &H3B4,1: OUT &H3B5,&H2D
[WB4] 1Ø5Ø OUT &H3B4,2: OUT &H3B5,&H2E
[YX3] 1Ø6Ø OUT &H3B4,3: OUT &H3B5,7
[FC4] 1Ø7Ø OUT &H3B4,4: OUT &H3B5,&H5B
[YX3] 1Ø8Ø OUT &H3B4,5: OUT &H3B5,2
[EC4] 1Ø9Ø OUT &H3B4,6: OUT &H3B5,&H57
[EB4] 11ØØ OUT &H3B4,7: OUT &H3B5,&H57
[BX3] 111Ø OUT &H3B4,9: OUT &H3B5,3
[256] 112Ø OUT &H3B8,1Ø ' Monitor einschalten
[7V1] 113Ø RETURN
```

Listing 4. »h-einsch.bas« schaltet den Grafikmodus der Hercules-Karte ein

```
[XY6] 999  ' Diese Routine stellt Text-Modus ein
[4C6] 1ØØØ OUT &H3BF,Ø  ' Grafik-Modus verbieten
[FU5] 1Ø1Ø OUT &H3B8,Ø  ' Monitor abschalten
[VG7] 1Ø2Ø ' Die nächsten 9 Zeilen setzen Text-Modus
[EB4] 1Ø3Ø OUT &H3B4,Ø: OUT &H3B5,&H61
[EB4] 1Ø4Ø OUT &H3B4,1: OUT &H3B5,&H5Ø
[NB4] 1Ø5Ø OUT &H3B4,2: OUT &H3B5,&H52
[CC4] 1Ø6Ø OUT &H3B4,3: OUT &H3B5,&HF
[PB4] 1Ø7Ø OUT &H3B4,4: OUT &H3B5,&H19
[YX3] 1Ø8Ø OUT &H3B4,5: OUT &H3B5,2
[ØC4] 1Ø9Ø OUT &H3B4,6: OUT &H3B5,&H19
[ØB4] 11ØØ OUT &H3B4,7: OUT &H3B5,&H19
[TB4] 111Ø OUT &H3B4,9: OUT &H3B5,&HD
[946] 112Ø OUT &H3B8,8  ' Monitor einschalten
[7V1] 113Ø RETURN
```

Listing 5. »h-textmo.bas« ist eine Basic-Routine, welche die Hercules-Karte für den Textmodus programmiert

```
[9H8] 1 'MASKEN% enthält die Verschiebungen für die Spa
    lten
[X82] 1Ø DIM MASKEN%(8)
[FA4] 2Ø MASKEN%(Ø)=128: MASKEN%(1)=64
[1B4] 3Ø MASKEN%(2)=32: MASKEN%(3)=16
[B14] 4Ø MASKEN%(4)=8: MASKEN%(5)=4
[RØ4] 5Ø MASKEN%(6)=2: MASKEN%(7)=1
[BP6] 9999  'Diese Routine schreibt einen Punkt
[EE8] 1ØØØØ IF (R%<Ø) OR (S%<Ø) OR (R%>347) OR (S%>719)
     THEN RETURN
[596] 1ØØ1Ø ADRS = (R% AND 3)*8192 + (R%\4)*9Ø + (S%\8)
[MM3] 1ØØ2Ø M% = MASKEN%(S% AND 7)
[5Y2] 1ØØ3Ø DEF SEG = &HBØØØ
[GP9] 1ØØ4Ø POKE ADRS,(PEEK(ADRS) AND (NOT M%)) OR (F%
    * MASKEN%(S% AND 7))
[7S1] 1ØØ5Ø DEF SEG
[KV1] 1ØØ6Ø RETURN
```

Listing 6. Direkten Zugriff auf den Bildschirmspeicher im Grafikmodus gestattet »h-pktsch.bas«: Diese Basic-Routine erlaubt das Schreiben eines Punktes

Die Programmierung der CGA-Karte

Die Programmierung der CGA-Karte (Color Graphics Adapter) muß kein Geheimnis sein. Anhand weniger Prinzipien und einer Menge praktischer Beispiele wird der Umgang mit dieser Grafikkarte vorgestellt, so daß Sie alle Merkmale der CGA-Karte voll ausnützen können, auch wenn diese nicht von einer Hochsprache unterstützt werden.

Die CGA-Karte ist eine der meistverbreiteten Grafikkarten im PC-Bereich, weil diese Karte sowohl mehrere Farben als auch Grafik zu einem sehr günstigen Preis bietet. Da die CGA-Karte seit langer Zeit verfügbar ist, wird sie von beinahe allen Hochsprachen unterstützt. Manchmal ist es aber wünschenswert, die Ausgabe auf diese Karte selbst vorzunehmen – sei es wegen der Geschwindigkeit (die Routinen mancher Hochsprachen sind sehr langsam) oder weil irgendwelche speziellen Effekte erzielt sein wollen.

Die CGA-Karte läßt sich in mehreren Modi betreiben. Der Betriebsmodus legt drei Eigenschaften der Bildschirmanzeige fest: Es gibt reine Textmodi, in denen sich keine Grafik darstellen läßt, und es gibt Grafikmodi, die sowohl Text als auch Grafik auf dem Bildschirm ausgeben können. Der jeweils gewählte Modus bestimmt auch, wie viele Farben höchstens gleichzeitig gezeigt werden. Selbst die effektive Grafikauflösung wird durch die Festlegung eines Bildschirmmodus gesteuert. In Tabelle 1 sind die verschiedenen Modi der CGA-Karte zusammengefaßt. Die Nummer in der linken Spalte wird als Kennzeichen für den Modus verwendet.

Die Rede ist dann etwa vom »Modus 6« oder »Modus 1«. Die Spalten »Art«, »Auflösung« und »Farben« beschreiben kurz jeden Modus.

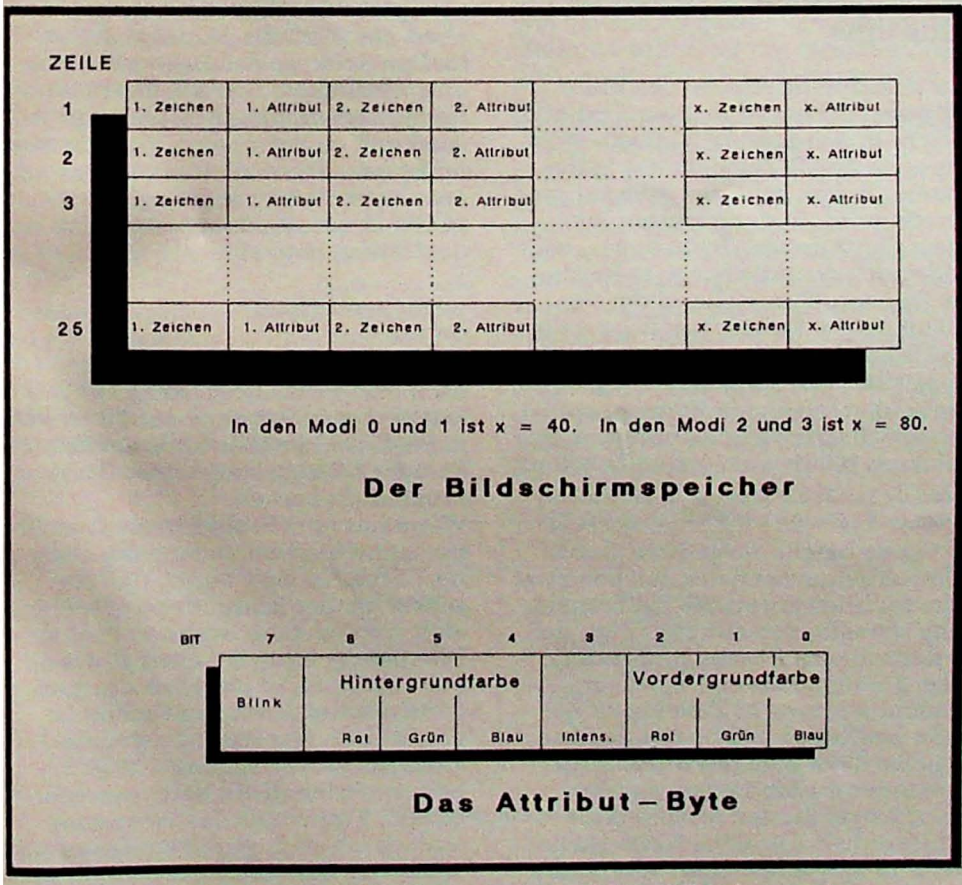


Bild 1. Der Bildschirmaufbau im CGA-Textmodus erfolgt mit Zeichen- und Attribut-Byte

Die CGA-Karte hat vier Textmodi in zwei Paaren organisiert. Zwei Modi stellen Text in 40 Spalten mal 25 Zeilen dar, die anderen beiden Modi verwirklichen die übliche Darstellung in 80 Spalten zu 25 Zeilen. In den Textmodi stehen 16 Farben zur Verfügung.

Außerdem bietet die CGA-Karte zwei Grafikmodi, die 320 (waagrecht) zu 200 (senkrecht) Punkte in vier Farben oder 640 zu 200 Punkte in zwei Farben auflösen.

Wer die Tabelle etwas näher betrachtet, merkt, daß einige Modi sich sehr ähnlich sind. Es gibt die drei paarigen Modi, null und eins, zwei und drei sowie vier und fünf, die sich nur in der Wiedergabe der Farben unterscheiden. Die Erklärung dieses etwas seltsamen Tatbestandes liegt in der Hardware begründet – in diesem Fall an der Grafikkarte und dem Monitor. Als die CGA-Karte erstmals auf den Markt kam, waren Farbmonitore drei- bis fünfmal so teuer, wie sie es heute sind. Deshalb hat die Original-CGA-Karte zwei Monitor-Ausgänge, einen für einen Farb- und einen für einen Monochrom-Monitor. Wer Grafik wollte, aber für einen Farbmonitor nicht mehrere tausend Mark hinblättern konnte, dem war somit trotzdem geholfen. Er konnte nämlich einen günstigen Monochrom-Monitor an die Farbgrafikkarte anschließen und damit die Grafikfähigkeit der CGA-Karte ausnützen – allerdings ohne Farbe. Dies funktionierte aber nur dann, wenn die CGA-Karte kein Farbsignal an den Monochrom-Monitor weitergab. Dies ist der Grund für die drei farblosen Modi.

Die CGA-Karte kann Text in 16 Farben darstellen: Zu den drei Farbkomponenten gesellt sich noch die Intensität

Die CGA-Karte kann 16 Farben ausgeben. Das Signal zum Monitor wird aus vier Komponenten zusammengestellt, wobei jede Komponente entweder in das Signal eingemischt oder aus dem Signal ausgelassen wird. Vier Komponenten mit je zwei möglichen Zuständen ergeben $2^4 = 16$ komplette Signale, die jeweils eine Farbe wiedergeben. Drei der vier Komponenten sind die Farbkomponenten Rot, Blau und Grün. Die vierte Komponente heißt »Intensität« und ist etwa als Helligkeit zu verstehen. Wenn die Intensitäts-Komponente aktiv ist, erhalten Sie einfach eine hellere Version der gleichen Grundfarbe auf dem Monitor. Die Farben können deshalb in die beiden Gruppen »hell« und »dunkel« aufgeteilt werden (Ta-

belle 2). Wie die Farben tatsächlich aussehen, hängt sehr stark vom Monitor ab. Manche Monitore können überhaupt nichts mit »Intensität« anfangen. Das heißt, die dunkle und die helle Version einer Farbe sehen gleich aus, während andere Monitore Schwierigkeiten mit manchen Farben (typischerweise Braun, Gelb und Grau) haben. Was bei Ihrem Monitor der Fall ist, können Sie nur durch Ausprobieren feststellen.

Tabelle 3 zeigt diejenigen Funktionen in GW-Basic, die den Bildschirmmodus setzen. Es ist kein Zugriff auf das BIOS oder die Hardware notwendig, weil alle Modi bequem und einfach von der Hochsprache aus gesetzt werden können. Basic bietet jedoch keine einfache Methode, um den Modus abzufragen. Hierzu könnten Sie die BIOS-Routine »Modus abfragen« direkt aufrufen, jedoch gibt es einen noch einfacheren Weg. Das BIOS speichert die Nummer des Modus nämlich an einer festen Adresse, 449hex. Mit den Basic-Zeilen

```
def seg = &h40
modus% = peek (&h49)
def seg
```

bekommen Sie den aktuellen Modus in der Variablen »Modus%« zurück.

Der Bildschirmspeicher wird auf zwei Arten verwendet. In den Textmodi werden ASCII-Codes in den Bildschirmspeicher geschrieben, und die CGA-Karte wandelt diese Codes in Punktmuster um, die wiederum als Zeichen auf dem Bildschirm gezeigt werden. In den Grafikmodi werden Punkte in Form verschlüsselter Farben in den Bildschirmspeicher geschrieben. Die CGA-Karte mischt das Farbsignal gemäß dem Speicherinhalt zusammen, muß aber keine weitere Umwandlung vornehmen.

Der Bildschirmspeicher ist ein reservierter Bereich im Gesamtspeicher des PC, der sowohl vom Prozessor (das heißt, vom Programm) als auch von der Bildschirmsteuerlogik gelesen und geschrieben werden kann. Dieser Speicher ist physikalisch getrennt vom »normalen« Arbeitsspeicher, in dem Programme und Daten abgelegt werden, und diese Trennung spiegelt sich auch in den Adressen wider. Während der Arbeitsspeicher eines PC von »0000:0000« bis maximal »9000:FFFF« liegt, ist der Bildschirmspeicher der CGA-Karte zwischen Adresse »B800:0000« und »B800:7FFF« angesiedelt und damit 16 KByte groß.

Modus	Art	Auflösung	Farben
0	Text	40x25	2
1	Text	40x25	16
2	Text	80x25	2
3	Text	80x25	16
4	Grafik	320x200	4
5	Grafik	320x200	4
6	Grafik	640x200	2

Tabelle 1. Die Betriebsarten (Modi) der CGA-Karte

Der Basisaufbau des Bildschirmspeichers in den Textmodi ist im oberen Teil von Bild 1 zu sehen. Wie dort abgebildet, besteht jedes angezeigte Zeichen aus zwei Byte. Das erste Byte ist einfach das Zeichen als ASCII-Wert, während das zweite Byte das sogenannte Attribut darstellt. Die Zusammenstellung dieses Attributs ist im unteren Teil des Bildes zu finden. Das Attribut bestimmt mit vier Bit die Farbe des Zeichens und mit drei Bit die Farbe des Hintergrunds. Das letzte Bit bestimmt, ob das Zeichen blinkend dargestellt sein soll. Das kleine Basic-Programm »vorf.bas« in Listing 1 führt die verschiedenen Attribut-Bits vor (Bild 5). Starten Sie das Programm und beachten Sie die Ausgabe, um einen schnellen Blick auf die Möglichkeiten der Attribute zu bekommen.

Die Adresse des zugehörigen Attribut-Byte ist immer die Adresse des Zeichens plus eins

Der prinzipielle Aufbau des Bildschirmspeichers ist in allen Textmodi gleich. In Adresse »B800:0000« steht das erste ASCII-Zeichen, das in der oberen, linken Ecke des Bildschirms gezeigt wird. In der nächsten Speicherzelle, Adresse »B800:0001«, befindet sich das Attribut-Byte für dieses Zeichen. Eine Zelle weiter, in »B800:0002«, steht schon das zweite ASCII-Zeichen, danach das zugehörige Attribut und so geht es weiter.

Die beiden folgenden Formeln ergeben jeweils die Verschiebungen eines Zeichens relativ zum ersten Zeichen:

```
Modus 0, 1: (Zeile x 80) + (Spalte x 2)
Modus 2, 3: (Zeile x 160) + (Spalte x 2)
```

Es wurde bereits ausgeführt, daß in einem Textmodus jedes Zeichen zwei Byte im Bildschirmspeicher braucht. Eine schnelle Berechnung zeigt, daß in den »hochauflösenden« Modi 2 oder 3 weniger als vier KByte gebraucht werden: 25 Zeilen x 80 Zeichen pro Zeile x 2 Byte pro Zeichen ergeben 4000 Byte pro Bildschirm. Im Modus 0 oder 1 genügen sogar nur 2 KByte für den ganzen Bildschirminhalt. Die CGA-Karte hat jedoch 16 KByte Speicher. Was ist mit dem Rest? Nun, der restliche Speicher ist nicht verloren. Es werden sogenannte Bildschirmseiten angeboten, die jeweils einen kompletten Bildschirm darstellen. Natürlich ist nur eine Seite auf einmal sichtbar, jedoch kann mit ein bißchen Mühe eine beliebige Seite zu jedem beliebigen Zeitpunkt geschrieben werden. Da die Modi 0 und 1 etwa zwei KByte pro Bildschirmseite beanspruchen, werden acht Seiten (mit Nummern von 0 bis 7) eingerichtet, während in den Modi 2 und 3 nur vier Seiten (Nummern von 0 bis 3) zur Verfügung stehen. Das BIOS bietet eine Funktion an, mit der die aktive Seite gesetzt werden kann. »Aktiv« bedeutet hier, daß diese Seite auf dem Bildschirm ausgegeben wird. Die anderen Seiten können nach wie vor angesprochen werden, sind aber unsichtbar, bis sie als neue aktive Seite gesetzt werden. Diese BIOS-Funktion ist auch in GW-Basic als Teil des Screen-Befehls zu finden. Eine Anweisung wie

```
screen ,,0,1
```

bewirkt, daß die Seite 1 auf dem Bildschirm gezeigt wird, während alle Print-Befehle ihre Ausgabe auf Seite 0 schreiben. Die allgemeine Form dieser Anweisung ist

```
screen ,,[Seite für Print],[sichtbare Seite]
```

Eine Funktion, die sowohl im BIOS als auch in den höheren Programmiersprachen fehlt, ist »aktive Seite abfragen«. Glücklicherweise speichert das BIOS die Nummer der aktuellen Seite, so daß Sie nur von der entsprechenden Adresse ein Byte ablesen müssen, um diese wichtige Auskunft zu bekommen. Die Adresse lautet hexadezimal »0040:0062« und kann sehr einfach abgefragt werden. In GW-Basic sind hierzu lediglich drei Zeilen notwendig:

```
def seg = &h40
seite% = peek(&h62)
def seg
```

Übrigens ist der Befehl »def seg« nach dem Peek-Befehl reine Höflichkeit und setzt das Segment für weitere Peek- und Poke-Befehle auf das Standard-Datensegment des Basic-Programms zurück.

Wenn nur eine Bildschirmseite sichtbar ist, wozu dienen dann die anderen? Eine Antwort lautet, daß ein sehr schneller Bildschirmwechsel erzielt werden kann, wenn das neue Bild bereits fertig auf einer anderen Seite steht. Es ist lediglich die neue Seite einzuschalten, und schon ist das Bild da. Das Basic-Programm »seitenw.bas« in Listing 2 zeigt Ihnen, wie schnell ein Seitenwechsel erfolgt. Tippen Sie das Programm einfach ein, und probieren Sie es aus – es wird Sie überzeugen.

Das Schreiben auf dem Bildschirm ist Bestandteil jeder Sprache. Jedoch ist

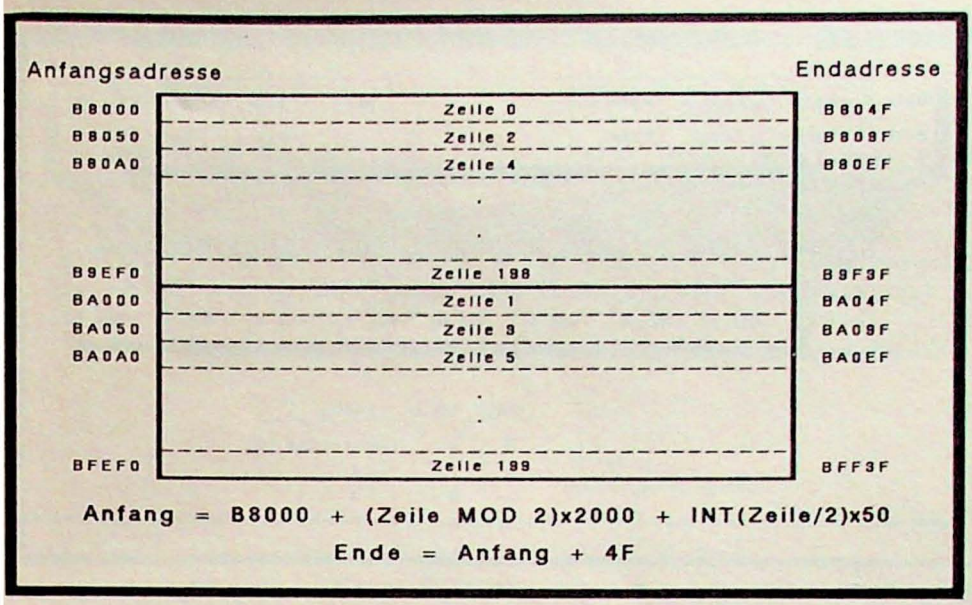


Bild 2. Der logische Aufbau des Bildschirmspeichers

es von einer Hochsprache aus umständlich, wenn Sie Farbe, Position (Zeile und Spalte) und Seite auf einmal setzen wollen. Aber auf der »niedrigen« Ebene der Hardware ist dies nicht schwierig.
Zunächst die Frage danach, was geschieht, wenn Sie in ASCII-Zeichen direkt in den Bildschirmspeicher schreiben. Wenn Sie die oben aufgeführten Formeln beachten, erscheint das Zeichen erfreulicherweise auf dem Bildschirm. Das gleiche gilt für Attribute: Schreiben Sie ein Attribut-Byte in den Bildschirmspeicher, bekommt das zugehörige Zeichen sofort die entsprechenden Eigenschaften. Doch wie ist im einzelnen vorzugehen?
Es werden zuerst die beiden Formeln erweitert, welche die Speicheradressen aus Spalte und Zeile berechnen, so daß auch die Seite berücksichtigt wird:

```
in Modus 0, 1: (Seite x 2048) + (Zeile x 80) + (Spalte x 2)
in Modus 2 oder 3: (Seite x 4096) + (Zeile x 160) + (Spalte x 2)
```

Wie vorher ist die Adresse des Attribut-Byte stets die Adresse des Zeichens plus eins. Jetzt sind alle Voraussetzungen für eine allgemeine Zeichenausgaberoutine gegeben. Obwohl GW-Basic dies kann, ist die angegebene Methode einfacher. Deswegen ist die Routine »ausg.bas« in Listing 3 Ihre Aufmerksamkeit wert: In Basic werden hier Zeichen und Attribute beliebig ausgegeben.

Eine ebenso sinnvolle Funktion ist das Lesen eines Zeichens von einer bestimmten Bildschirmposition. Obwohl GW-Basic diese nützliche Funktion als Screen-Befehl bietet, geht es wie bei der Ausgabe auch hier einfacher und schneller mit direkten Zugriffen auf den Bildschirmspeicher.

Da die Routine praktisch identisch ist mit der Ausgaberoutine, erübrigt sich eine weitere Erklärung. In Listing 4 findet sich die Basic-Routine »ascattr.bas«, die das gelesene ASCII-Zeichen in der Variablen »ZCode« und das gelesene Attribut in der Variablen »ZAttr« zurückgibt. Damit sind die wichtigsten Funktionen in den Textmodi schon angesprochen. Alle Funktionen liefern in allen Modi das erwünschte Ergebnis. Achten Sie lediglich darauf, daß Seite, Zeile und Spalte stimmen.

Rot	Grün	Blau	Nr.	Farbe	Nr.	Farbe
0	0	0	0	Schwarz	8	Dunkelgrau
0	0	1	1	Blau	9	Hellblau
0	1	0	2	Grün	10	Hellgrün
0	1	1	3	Türkis	11	Helltürkis
1	0	0	4	Rot	12	Hellrot
1	0	1	5	Violett	13	Lila
1	1	0	6	Braun	14	Gelb
1	1	1	7	Hellgrau	15	Weiß

Farbkomponente	
Intensität = 0	Intensität = 1
(dunkel)	(hell)

Tabelle 2. Die Farben der CGA-Karte

Programmierer aufgepaßt!

DMV bringt jetzt:

Das Buch zu BASIC2

PC1512/1640-Besitzer können aufatmen. Jetzt ist sie da, die BASIC2-Toolbox. Mit diesem einzigartigen Buch sind Sie nun in der Lage, das Locomotive BASIC2 besser auszunutzen, die Programmierung wird zum Kinderspiel.
Die BASIC2-Toolbox bietet nicht die x-te Einführung, sondern gibt anhand leistungsstarker und praxisorientierter Routinen und Programme den nötigen Durchblick.
Schritt für Schritt lernen Sie die strukturierte Programmentwicklung bis hin zum fertigen und lauffähigen Programm. Der Text ist in leicht und für jedermann verständlicher Form geschrieben, der Lerneffekt ist quasi garantiert. Alle Beispielprogramme sind sofort nachvollziehbar und stammen aus der täglichen Anwendungspraxis.

Einige Beispiele aus dem Inhalt:

Kurze Einführung in die Software-Entwicklung
- Grundlagen und Struktogramme

Werkzeuge für BASIC2
- Lister ermöglicht die formatierte Druckausgabe für BASIC2-Programme
- COMP komprimiert Ihre BASIC2-Programme
- Preprozessor für Include-Dateien
- Cross ist ein Generator zur Erzeugung von Querverweislisten

Werkzeuge zur Behandlung von Textdateien
- Auswertung von Word-Textdateien
- CUT, ein Filter für Textdateien
- PASTE,Vereinigung von Textdateien

Werkzeuge zur Software-Entwicklung
- CALC, ein Rechner für verschiedene Zahlensysteme
- DUMP, Ausgabe beliebiger Dateien im Hexformat.
- COMHEX, Umwandlung von COM-Dateien in HEX-Files

Ein kompletter Disassembler für den Intel 8086
- Von der Struktur zum fertigen Programm. Ein Disassembler wird programmiert.

Die BASIC2-Toolbox

Autor: Günter Born,
250 Seiten, 54 Abb.,
ISBN Nr. 3-926177-01-2 (Best.-Nr. 402)

Inland:		Ausland:	
Einzelpreis	49,– DM	Einzelpreis	49,– DM
zzgl. Versandkosten	4,– DM	zzgl. Versandkosten	6,– DM
Endpreis	53,– DM	Endpreis	55,– DM

Zu beziehen über den Computerfachhandel, den guten Fachbuchhandel oder direkt beim Verlag. Händleranfragen erwünscht.

Bitte benutzen Sie unsere Bestellkarte

DMV-Verlag

Postfach 250 · 3440 Eschwege

Zuletzt soll eine Gruppe von Funktionen betrachtet werden, die mit dem Cursor in Verbindung steht: So setzt eine Funktion die Position des Cursor, bewegt also den Cursor zu der gewünschten Stelle, und eine zweite fragt die Position des Cursor ab. Beide sind bereits in Basic zu finden: »Locate« setzt die Cursor-Position, und »Csrlin« und »Pos« geben die aktuelle Cursor-Position zurück.

Cursor-Funktionen auf allen Bildschirmseiten arbeiten nur mit Hilfe des BIOS

Wie üblich, weisen die Routinen der Hochsprache eine Schwäche auf – sie arbeiten nämlich nur auf der aktuellen Seite. Das heißt, eine Bildschirmseite muß sichtbar sein, um den Cursor bewegen zu können. Natürlich geht es aber auch anders, und der andere Weg ist nicht nur effektiver, sondern auch schneller. Es funktioniert wieder über das BIOS, das erfreulicherweise die Cursor-Positionen auf allen Seiten speichert. An der Adresse 450hex beginnt ein Array von 16 Byte – in acht Paaren organisiert. Jedes Paar besteht aus einer Spaltenangabe im ersten Byte und einer Zeilenangabe im zweiten Byte, die zusammen die aktuelle Cursor-Position auf der Seite abbilden. Das Lesen dieser Bytes ersetzt völlig die Funktion »Cursor-Position abfragen«, und das Schreiben ersetzt mit einer Ausnahme die Funktion »Cursor-Position setzen«. Die Ausnahme besagt, daß die neu geschriebene Cursor-Position erst dann in Kraft tritt, wenn zum Bildschirm geschrieben wird, während die Funktion »Cursor setzen« sofort wirkt. Wenn die Variable »Seite%« mit der Nummer der gewünschten Seite belegt ist, wird die Cursor-Position in GW-Basic so gelesen:

```
def seg = &h40
zeile% = peek (&h50 + seite% * 2 + 1)
spalte% = peek (&h50 + seite% * 2)
def seg
```

Die letzte in diesem Zusammenhang vorzustellende Routine bestimmt die Größe des Cursor. Um diese Routine zu verwenden, muß bekannt sein, wie ein Zeichen auf dem Bildschirm aus Punkten zusammengestellt wird. Die Punktmatrix besteht bei der Farbgrafikkarte aus acht Reihen von je acht Punkten. Die Reihen werden von oben (Reihe 0) nach unten (Reihe 7) numeriert. Mit der BIOS-Routine »Cursorgröße setzen« wird festgelegt, aus welchen Reihen der Cursor besteht. Diese Routine ist in Basic bereits als Teil des Locate-Befehls implementiert. Zum Beispiel erzeugt der Befehl

```
locate ,,,7,7
```

den bekannten Unterstrich als Cursor, während der Befehl

```
locate ,,,0,7
```

einen Block als Cursor ausgibt. Der Cursor kann in Basic problemlos ausgeschaltet werden, so daß er unsichtbar ist:

```
locate ,,0
```

schaltet den Cursor ab, und

```
locate ,,1
```

schaltet ihn wieder ein. Es gibt keine Funktion, welche die aktuelle Größe des Cursor zurückgibt. Dabei bietet das BIOS wieder Hilfe. In den zwei Byte ab Adresse 460hex steht die Anfangs- und Ende-Reihe des Cursor. Seltsamerweise steht das Ende vor dem Anfang. Diese beiden Byte können analog zum Cursor-Position-Array gelesen werden, jedoch hat das Schreiben in diese Bytes keine sofortige Auswirkung.

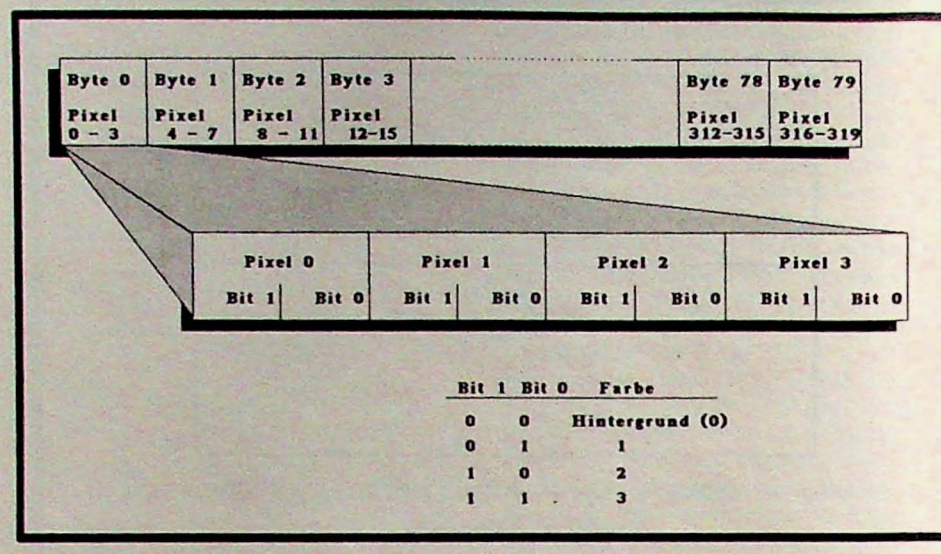


Bild 3. Die Speicherung einer Zeile im vierfarbigen CGA-Modus 4 erfolgt je Bildpunkt in zwei Bit

Der Bildschirmspeicher im Grafikmodus ist anders organisiert als im Textmodus

Wie bereits erwähnt, ist die Arbeitsweise der CGA-Karte im Grafikmodus wesentlich anders als in einem Textmodus. Dieser Unterschied läßt sich auch am logischen Aufbau des Bildschirmspeichers ablesen (Bild 2). Zunächst sieht der Speicher ganz »ordentlich« aus – die ersten 80 Byte enthalten die erste Grafikpunktzeile (Zeile Nummer 0). Aber dann müssen Sie die zweite Zeile suchen – und suchen – und suchen... Die zweite Grafikzeile (Zeile Nummer 1) steht nämlich nicht direkt nach der ersten, wie in den Textmodi, sondern am Anfang der zweiten Hälfte des Bildschirmspeichers, genauer: an der Adresse »B800:2000«. Aber was steht dann nach der ersten Zeile? Es ist die dritte Zeile! Und nach der zweiten folgt die vierte, nach der dritten die fünfte, und so abwechselnd weiter, bis endlich alle 200 Zeilen im Speicher stehen. Der Bildschirmspeicher ist also in den Grafikmodi in zwei Abschnitte geteilt. Die erste Hälfte beinhaltet alle geraden Zeilen (Nummern 0, 2, 4 etc.), eine nach der anderen, und die zweite Hälfte enthält alle ungeraden Zeilen (1, 3, 5 etc.) in der gleichen Reihenfolge. Auf die Frage, warum der Speicher so (un)organisiert wurde, bekommt man eine leider allzu übliche Antwort: Die Hardware der CGA-Karte wurde dadurch ein wenig einfacher. Daß die Programmierung der Hardware da-

Palette	Farbe 0	Farbe 1	Farbe 2	Farbe 3
0	(*)	Grün	Rot	Braun
1	(*)	Türkis	Magenta	Hellgrau
2	(*)	Hellgrün	Hellrot	Gelb
3	(*)	Helltürkis	Lila	Weiß

(*) Farbe 0 ist standardmäßig Schwarz, kann aber auf jede beliebige Farbe gesetzt werden.

Tabelle 3. Die Farbpaletten der CGA-Karte

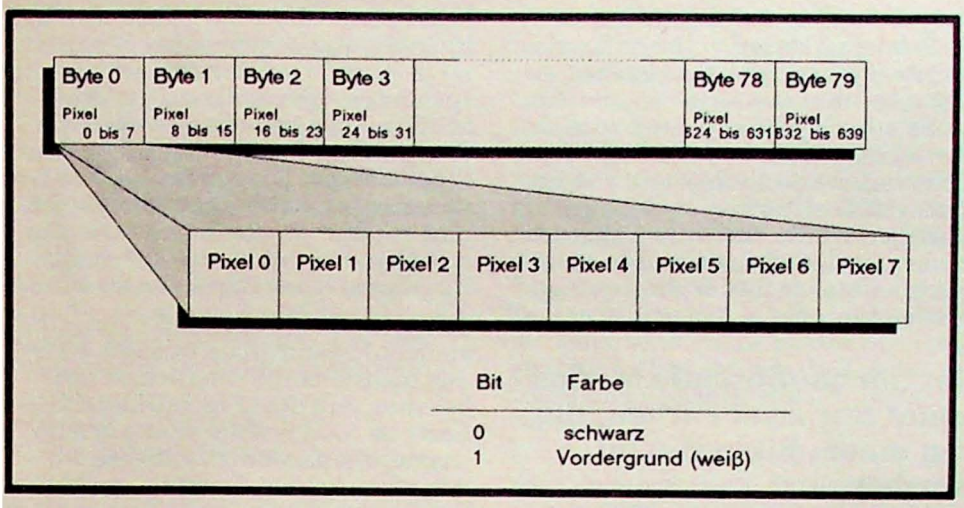


Bild 4. Der Aufbau einer Zeile des hochauflösenden CGA-Modus 6

durch wesentlich erschwert wird, und viele Programme dadurch sehr verlangsamt werden, war den Hardware-Entwicklern dabei gleichgültig. Für alle Grafikmodi ist diese Grundstruktur gleich, und zusätzlich ist eine Grafikzeile am Bildschirm immer 80 Byte lang. Eine Formel für die Anfangsadresse einer Grafikzeile entwickelt sich demgemäß:

```
(Zeile mod 2)x8192 + (int(Zeile/2)x80)
```

Da die anderen Eigenschaften des Speichers und die Verwendung der BIOS-Routinen stark vom gewählten Modus abhängen, werden zunächst die Modi 4 und 5 (320x200 Bildpunkte in vier Farben beziehungsweise Graustufen) untersucht. Die Vorstellung des hochauflösenden Modus 6 schließt sich an.

Der vierfarbige Modus 4 der CGA-Karte ist der beliebteste und meistverwendete Grafikmodus. Dieser Modus bietet eine relativ hohe Auflösung in mehreren Farben, was kein anderer Modus der CGA-Karte bietet. Er wird deswegen in allen Bereichen eingesetzt. Wie eine Zeile in diesem Modus gespeichert wird, sehen Sie in Bild 3. Jeder Punkt auf dem Bildschirm wird in zwei Bit abgebildet, deren Wert die Farbe des Punktes bestimmt. Deshalb gibt es in diesem Grafikmodus vier mögliche Farben.

GW-Basic kann auch im Grafikmodus direkt auf den Bildschirmspeicher der CGA-Karte zugreifen

Es werden genau vier Punkte in jedem Byte gespeichert: die vier linken Punkte einer Zeile im ersten Byte, die nächsten vier im zweiten Byte und so weiter. Bild 3 zeigt auch, wie die vier Punkte innerhalb eines Byte abgelegt werden. Damit kann die Formel für die Adresse einer Zeile erweitert werden, um die genaue Position eines Punktes zurückzugeben. Hier reicht die Adresse nicht aus, da Sie auch wissen müssen, in welchen beiden Bits der Punkt gespeichert wird. Diese zusätzliche Information berechnet sich als ein Byte, in dem genau zwei Bit gesetzt sind, und zwar die beiden, die den gesuchten Punkt abbilden. Dieses Byte wird »Maske« genannt. Die Formel für die Adresse lautet:

```
adr% = (zeile% and 1)*8192 + (zeile%\2)
*80 + (spalte%\4)
```

Weil die Berechnung der Maske relativ viel Zeit in Anspruch nimmt, geht es am schnellsten, die Verschiebungen für die Masken in Variablen zu speichern und den entsprechenden Wert aus der Variablen zu lesen:

```
dim masken%(4)
masken%(0) = 64: masken%(1) = 16
masken%(2) = 4: masken%(3) = 1
maske% = 3 * masken%(spalte% and 3)
```

Mit diesen Auskünften können Sie bereits einige Programmzeilen finden, die einen direkten Zugriff auf den Bildschirmspeicher auch im Grafikmodus zulassen. Es gibt auch BIOS-Routinen, um Bildpunkte zu schreiben und zu lesen, weil aber Basic diese Funktionen (»Pset«, »Preset« und »Point«) bereits beinhaltet, sind diese BIOS-Routinen in diesem Zusammenhang zu vernachlässigen.

Eine Basic-Routine zum Schreiben eines Punktes finden Sie in Listing 5 (»punkt.bas«).

Die Berechnung der Adresse und Maske ist bereits bekannt, und die Farbe wird analog zur Maske berechnet. In der fünften Zeile der Routine wird das Byte aus dem Bildschirmspeicher gelesen, um die anderen drei Punkte wieder schreiben zu können. Mit »M%« löschen Sie die beiden Bits, die zu dem zu schreibenden Punkt gehören, und danach setzen Sie die Bits, um die gewünschte Farbe zu erhalten. Das Ganze wird sofort wieder in den Bildschirmspeicher geschrieben, und die Routine kehrt zurück.

Wenn ein Punkt nur gelesen werden soll, ist ähnlich vorzugehen. Adresse und Maske werden gleich berechnet, die Farbe fällt weg – sie ist nämlich gesucht. Wenn Sie die Farbe ermittelt haben, müssen Sie lediglich die beiden Bit ganz nach rechts bringen, um eine Zahl von 0 bis 3 zurückzuliefern. Den Vorgang können Sie nach dem Beispiel von Listing 5 programmieren; in der neuen Routine ist lediglich die Verschiebung der Farbe anders als in »punkt.bas«. Die Formeln für Adresse und Maske bleiben natürlich gleich.

Eine schnelle Berechnung zeigt, daß in diesen beiden Modi der Bildschirmspeicher praktisch voll ausgenutzt wird. Es sind 320x200 = 64000 Punkte; geteilt durch 4 Punkte pro Byte, ergibt das 16000 Byte, um ein Bild zu speichern. Von den 16 KByte auf der CGA-Karte sind also nur noch 356 Byte übrig – und die werden nicht verwendet.

Jetzt wissen Sie, daß jeder Punkt in Form von zwei Bit gespeichert wird und wie Sie die Bits im Bildschirmspeicher finden können. Aber wie bestimmen die Bits die Farbe des Punktes, und welche vier Farben von den 16 werden verwendet?

Das Farbpalettenkonzept regelt die Farbauswahl: Mittlerweile stehen vier Paletten zur Wahl

Diese Fragen werden durch ein neues Konzept, die Farbpalette, beantwortet. Die Farbpalette beinhaltet, wie bei einem Maler, eine Auswahl aller zur Verfügung stehenden Farben. Bei der Original-IBM-Farbgrafikkarte gibt es zwei Paletten. Dies sind die Paletten 0 und 1 in Tabelle 3. Inzwischen können fast alle anderen CGA-Karten mit einer von vier Paletten arbeiten, wobei die beiden neuen (die Paletten 2 und 3 in der Tabelle) die hellen Versionen der Original-Farbpaletten sind. »palet.bas«, ein Basic-Programm zur Demonstration der Farbpaletten (Bild 6), finden Sie in Listing 6. Wenn vier verschiedene Ergebnisse zu sehen sind, besitzen Sie eine moderne CGA-Karte. Dies verdeutlicht nebenbei eine interessante Eigenschaft der Paletten: Wenn die Palette gewechselt wird, wechseln sofort alle Farben auf dem Bildschirm. Eine weitere Eigenschaft aller Paletten stellt die Hintergrundfarbe dar. Diese Farbe hat immer die Nummer 0, es kann aber eine beliebige Farbe zur Hintergrundfarbe erklärt werden.

Das heißt, in jeder Palette sind drei Farben festgelegt, während die vierte

frei wählbar ist. Diese »freie« Farbe ist aber, wie gesagt, die Hintergrundfarbe. Ein gelöschter Bildschirm zeigt stets diese Farbe.
Die Palette können Sie von Basic aus setzen. Dafür brauchen Sie nur eine einzige Anweisung, zum Beispiel:

```
color 15,1
```

Hier ist »15« die (frei programmierbare) Hintergrundfarbe, während »1« die Nummer der Palette darstellt. Allerdings lehnt es Basic grundsätzlich ab, die Paletten 2 oder 3 zu setzen. Je nach Basic-Version wird eine Fehlermeldung ausgegeben oder die Anweisung (zum Beispiel »color 15,2«) einfach ignoriert. Es gibt aber einen genialen Umweg, der einen einzigen direkten Zugriff auf die Bildschirmsteuerlogik fordert. Er funktioniert wie folgt: Zunächst wählen Sie Ihre Hintergrundfarbe aus, danach die gewünschte Palette. Suchen Sie in der folgenden Tabelle die Palette aus, und addieren Sie den nebenstehenden Wert zu der Nummer Ihrer Hintergrundfarbe:

Palette	Wert
0	0
1	32
2	16
3	48

Setzen Sie das Resultat anstelle von »x« in der folgenden Basic-Anweisung ein:

```
out &h3d9,x
```

Mehr ist nicht zu tun. Die Farbpalette ist nach Wunsch eingestellt. Natürlich gibt es auch eine BIOS-Funktion, um die Palette zu setzen, aber sie ist nun bedeutungslos. Zur Abfrage der aktiven Palette bietet GW-Basic keine Funktion an, und sogar das BIOS schweigt an dieser Stelle. Nirgendwo in der PC-Literatur wird erwähnt, daß das BIOS, das MS-DOS oder sonst ein Utility die Palette irgendwo speichert. Nun könnte man vermuten, daß – wenn oben schon ein Ausgabebefehl verwendet wird, um die Palette zu setzen – der entsprechende Eingabebefehl in gleicher Weise verwendet werden könnte, um die Palette abzulesen. Leider stimmt dies nicht – jeder lesende Zugriff auf die Port-Adresse 3d9hex gibt 255 zurück, gleichgültig was vorher geschrieben wurde. Sie sollten sich also merken, welche Palette gerade ausgewählt wurde, da dies nicht abgefragt werden kann.

Der hochauflösende Modus kennt nur zwei Farben, die von einem Bit bestimmt werden

Der letzte Modus der CGA-Karte ist der hochauflösende Modus 6. Dabei besteht jede Zeile aus 640 Punkten, dem Doppelten der Modi 4 und 5. In diesem Modus wird jeder Punkt als ein Bit gespeichert; dies ergibt die gleiche Ausnutzung des Bildspeichers wie bei den anderen beiden Grafikmodi. Der Aufbau einer Zeile verläuft ähnlich zu den Modi 4 und 5 (Bild 4). Die Zeilenadressen sind sogar gleich, weil eine Zeile genau so viele Bits benötigt wie vorher; die Maske besteht jedoch nur noch aus einem einzigen Bit. Die Formel für die Byte-Adresse eines Punktes in Modus 6 lautet:

```
adr% = (zeile% and 1)*8192 +
(zeile%\2)*80 + (spalte%\8)
```

Diese Formel für die Maske ist wiederum zeitaufwendig; die Masken sind daher besser in Variablen aufgehoben:

```
dim masken%(8)
masken%(0) = 128: masken%(1) = 64
masken%(2) = 32: masken%(3) = 16
masken%(4) = 8: masken%(5) = 4
masken%(6) = 2: masken%(7) = 1
maske% = masken%(spalte% and 7)
```

Um Punkte zu lesen oder zu schreiben, müssen Sie wiederum entweder BIOS-Routinen verwenden oder direkte Zugriffe auf den Bildschirmspeicher programmieren. GW-Basic beherrscht das schon, und zwar mit den gleichen Funktionen wie in den Modi 4 und 5. Sie können die Routine »ascattr.bas« als Beispiel nehmen und mit den obigen Zeilen Routinen erstellen, um Punkte in Modus 6 zu schreiben beziehungsweise zu lesen.

Da es nur ein Bit pro Punkt gibt, sind auch nur zwei Farben in diesem Modus realisierbar. Sicherlich ist Ihnen bekannt, daß die Original-IBM-CGA-Karte im Modus 6 nur schwarz (Hintergrund) und weiß (Vordergrund) darstellen konnte. Inzwischen läßt sich jedoch die Vordergrundfarbe frei auswählen. GW-Basic und das BIOS bieten jedoch keine Möglichkeiten, um dies zu programmieren. Falls Sie sich einen neueren PC zugelegt haben, verwenden Sie die folgende Basic-Zeile, um im Modus 6 eine beliebige Vordergrundfarbe einzusetzen:

```
out &3D9,Farbe
```

»Farbe« kann jeden Wert von 0 bis 15 annehmen. Eine Umprogrammierung dieses Wertes stellt auf dem Bildschirm alles in der neuen Farbe dar, es ergibt sich mitunter ein schöner Effekt.

Grafik ist schön und gut, aber meist ist nicht die reine Grafik, sondern eine Mischung aus Grafik und Text gefragt. Wie läßt sich dies verwirklichen? Zunächst sollte bekannt sein, daß Befehle wie »print« in Basic auch in einem Grafikmodus funktionieren. Das heißt, von Basic aus lassen sich ohne weiteres Grafik und Text mischen. Die Farbe, in der der Text ausgegeben wird, ist vom Modus und gegebenenfalls von der Palette abhängig. Im Modus 4 ist die Textfarbe stets Farbe 3 der aktuellen Palette, im Modus 5 die entsprechende Graustu-

Bild 5. Die Farben auf einem CGA-Bildschirm, wie sie von Listing 1 erzeugt werden

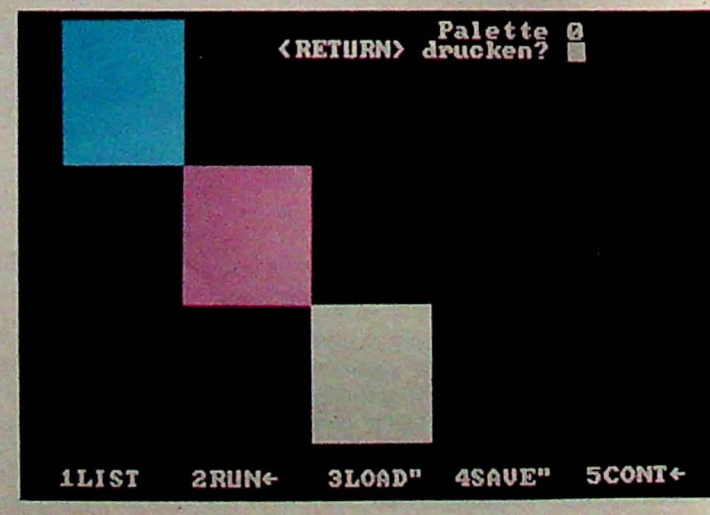


Bild 6. Auf diese Weise präsentiert Listing 6 die Farbpaletten auf Ihrem CGA-Monitor

fe. Die Textfarbe im Modus 6 ist logischerweise immer die Vordergrundfarbe.
Ein Problem beim Kombinieren von Text und Grafik entsteht dadurch, daß die Bildschirmkoordinaten für den Text und die Grafik unterschiedlich sind: Für Text finden nach wie vor Spalten und Zeilen in einer 40/80x25 Textmatrix Verwendung. Für eine Grafik setzt man dagegen die x- und y-Koordinaten des Grafikbildschirms ein, 320/640x200. Die Umwandlung ist jedoch nicht schwierig:

```
Textzeile = (Grafik-X/8) + 1
Textspalte = (Grafik-Y/8) + 1
```

Die Formeln gelten für alle drei Grafikmodi.
Es wurde schon gezeigt, wie ein Zeichen vom Bildschirm gelesen wird, wenn ein Textmodus aktiv ist. Wie aber funktioniert dies, wenn es sich bei dem aktuellen Modus um einen Grafikmodus handelt? Basic macht das Leben auch hier einfacher: Der Screen-Befehl gibt das Erwartete zurück, sowohl in einem Text- als auch in einem Grafikmodus. Dabei ist lediglich zu beachten, daß die Screen-Funktion auch in einem Grafikmodus Textkoordinaten braucht. Verwenden Sie daher einfach die beiden oben angeführten Formeln, um die Grafikkoordinaten gegebenenfalls umzuwandeln.

Das Interessante an dieser Funktion wird deutlich, wenn Sie daran denken, wie im Grafikmodus ein Zeichen im Bildschirmspeicher abgelegt wird. Das Zeichen wird nämlich, wie alles andere auch, als Punktraster gespeichert. Diese Routine muß dann ein Muster aus dem Bildschirm herauslesen und dieses Muster mit allen möglichen Zeichenmustern vergleichen, um das Zeichen zu identifizieren. Dies ist keine leichte Aufgabe. Basic, beziehungsweise das BIOS, erspart dem Programmierer also an dieser Stelle eine Menge Arbeit.

Damit haben Sie bereits die Grundlagen der CGA-Karte kennengelernt. Was hier geboten wurde, sollte Ihre Arbeit erleichtern und vor allem Anreiz bieten für Ihre eigenen Programme.

(Thomas Little/al)

```
[AW7] 1 ' Initialisierung - Modus 3, lösche Bildschirm
[783] 1Ø SCREEN Ø,1: WIDTH 8Ø
[8UØ] 2Ø CLS
[CØ4] 21 ' Schreibe Text-Muster
[ØØ2] 3Ø FOR I=6 TO 21
[DM3] 4Ø   FOR J=12 TO 72 STEP 4
[Ø74] 5Ø     LOCATE I,J: PRINT "ABC";
[CF1] 6Ø     NEXT
[W61] 7Ø   NEXT
[2C5] 71 ' Schreibe Attribut-Byte-Werte
[W73] 8Ø FOR I=13 TO 73 STEP 4
[245] 9Ø  LOCATE 4,I: PRINT HEX$((I-13)*4);
[VC1] 1ØØ   NEXT
[ØD2] 11Ø FOR I=6 TO 21
[Ø15] 12Ø   LOCATE I,3: PRINT HEX$(I-6);
[4D1] 13Ø   NEXT
[MØ3] 131 ' Schreibe Titel
[YZ7] 14Ø LOCATE 1,25: PRINT "Attribute in den Text-Mod
      i"
[NV4] 141 ' Fange mit Attribut = Ø an
[8N1] 15Ø ATTR = Ø
[A77] 151 ' Setze Bildschirmspeicher als POKE-Ziel
[UR2] 16Ø DEF SEG = &HB8ØØ
[KA9] 161 ' Schreibe die Attribute direkt in Bildschirm
      speicher
[KD2] 17Ø FOR I=6 TO 21
[GP3] 18Ø   FOR J=12 TO 72 STEP 4
[AA4] 19Ø     ADDR = (I-1)*16Ø + (J-1)*2 + 1
[FU8] 2ØØ     POKE ADDR,ATTR: POKE ADDR+2,ATTR: POKE AD
      DR+4,ATTR
[8V2] 21Ø     ATTR = ATTR+1
[TH1] 22Ø     NEXT
[6D1] 23Ø   NEXT
[347] 231 ' setze SEG zurück (vorsichtshalber...)
[ØS1] 24Ø DEF SEG
[ØG7] 25Ø LOCATE 24,25: PRINT "Ende mit [Ctrl-Break]";
[YP1] 26Ø GOTO 26Ø
```

Listing 1. Das Basic-Programm »vorf.bas« führt Ihnen die Attribut-Bits des CGA-Bildschirms vor (siehe Bild 5)

```
[6UØ] 1Ø CLS
[UØ2] 2Ø SCREEN ,,1,Ø
[Y24] 4Ø FOR I=Ø TO 1999: PRINT "1";: NEXT
[RPØ] 45 C=1
[YØ2] 5Ø SCREEN ,,Ø,Ø
[4F5] 55 C=((C+1) MOD 8)+1: COLOR C: LOCATE 1,1
[N82] 6Ø FOR I=Ø TO 1999
[635] 7Ø PRINT "Ø";: IF INKEY$<>"" THEN 1ØØ
[G51] 8Ø NEXT
[TD1] 9Ø GOTO 5Ø
[U52] 1ØØ SCREEN ,,1,1
[7E3] 11Ø IF INKEY$<>"" THEN 5Ø
[TN1] 12Ø GOTO 11Ø
```

Listing 2. »seitenw.bas« zeigt den Seitenwechsel des CGA-Bildschirms

```
[4Ø8] 1ØØØØ IF (Z<1) OR (SP<1) OR (S<Ø) OR (Z>24) THEN
      RETURN
[FE9] 1ØØ1Ø DEF SEG = &H4Ø: MM=PEEK(&H49): DEF SEG: IF
      MM>3 THEN RETURN
[P83] 1ØØ2Ø IF MM>1 THEN 1ØØ4Ø
[SZ7] 1ØØ3Ø MM=1: IF (S>7) OR (SP>4Ø) THEN RETURN ELSE
      1ØØ5Ø
[SG6] 1ØØ4Ø MM=2: IF (S>3) OR (SP>8Ø) THEN RETURN
[FY2] 1ØØ5Ø DEF SEG = &HB8ØØ
[Ø19] 1ØØ6Ø MM=S*MM*2Ø48+(Z-1)*MM*8Ø+(SP-1)*2: POKE MM,
      C: POKE MM+1,A
[9S1] 1ØØ7Ø DEF SEG
[MV1] 1ØØ8Ø RETURN
```

Listing 3. »ausg.bas« gibt Zeichen und Attribute eines CGA-Bildschirms beliebig aus (Routine)

```
[DL8] 9998 '»ascattr.bas«, eine Routine zur Ausgabe des
[TN6] 9999 ' Zeichens mit seinem Attribut-Byte
[YZ1] 1ØØØØ C=-1: A=-1
[5Ø8] 1ØØ1Ø IF (Z<1) OR (SP<1) OR (S<Ø) OR (Z>24) THEN
      RETURN
[GE9] 1ØØ2Ø DEF SEG = &H4Ø: MM=PEEK(&H49): DEF SEG: IF
      MM>3 THEN RETURN
[V83] 1ØØ3Ø IF MM>1 THEN 1ØØ5Ø
[VZ7] 1ØØ4Ø MM=1: IF (S>7) OR (SP>4Ø) THEN RETURN ELSE
      1ØØ6Ø
[TG6] 1ØØ5Ø MM=2: IF (S>3) OR (SP>8Ø) THEN RETURN
[GY2] 1ØØ6Ø DEF SEG = &HB8ØØ
[539] 1ØØ7Ø MM=S*MM*2Ø48+(Z-1)*MM*8Ø+(SP-1)*2: C=PEEK(M
      M): A=PEEK(MM+1)
[AS1] 1ØØ8Ø DEF SEG
[NV1] 1ØØ9Ø RETURN
```

Listing 4. »ascattr.bas« gibt das gelesene ASCII-Zeichen und das Attribut-Byte des CGA-Bildschirms zurück (Routine)

```
[4Ø8] 1 'MASKEN% enthält Verschiebungen für die Spalten
[T82] 1Ø DIM MASKEN%(4)
[3B4] 2Ø MASKEN%(Ø)=64: MASKEN%(1)=16
[BØ4] 3Ø MASKEN%(2)=4: MASKEN%(3)=1
[BP6] 9999  'Diese Routine schreibt einen Punkt
[KE8] 1ØØØØ IF (R%<Ø) OR (S%<Ø) OR (R%>199) OR (S%>319)
       THEN RETURN
[N86] 1ØØ1Ø ADRS = (R% AND 1)*8192 + (R%\2)*8Ø + (S%\4)
[SØ4] 1ØØ2Ø M% = 3 * MASKEN%(S AND 3)
[DY2] 1ØØ3Ø DEF SEG = &HB8ØØ
[ØP9] 1ØØ4Ø POKE ADRS,(PEEK(ADRS) AND (NOT M%)) OR (F%
      * MASKEN%(S% AND 3))
[7S1] 1ØØ5Ø DEF SEG
[KV1] 1ØØ6Ø RETURN
```

Listing 5. »punkt.bas« schreibt einen Punkt auf den CGA-Bildschirm (Routine)

```
[BM1] 1Ø DIM PAL(4)
[6F6] 2Ø PAL(Ø)=Ø: PAL(1)=&H2Ø: PAL(2)=&H1Ø: PAL(3)=&H3
      Ø
[AR1] 3Ø SCREEN 1,Ø
[BU2] 4Ø CLS: OUT &H3D9,Ø
[383] 5Ø LINE (Ø,Ø)-(59,59),1,BF
[BS3] 6Ø LINE (6Ø,6Ø)-(119,119),2,BF
[XZ3] 7Ø LINE (12Ø,12Ø)-(179,179),3,BF
[UPØ] 8Ø P=Ø
[GX4] 9Ø LOCATE 1,24: PRINT "Palette";P;
[8U6] 1ØØ LOCATE 2,14: INPUT "<RETURN> drucken";I$
[G46] 11Ø P=(P+1) MOD 4: OUT &H3D9,PAL(P): GOTO 9Ø
```

Listing 6. »palet.bas« ist ein Basic-Programm zur Demonstration der Farbpaletten (vergleiche Bild 6)

Die Welt im Computer

Erweitern Sie Ihre Geographie-Datenbank mit Karten der jeweiligen Länder, geben Sie einem selbstprogrammierten Spiel den letzten grafischen Schliff, oder tragen Sie Ihre Umsatzzahlen zur Präsentation direkt in die grafische Darstellung der einzelnen Länder ein: alles, was Sie dazu benötigen, ist eine Weltkarte.

Wege, die Länder dieser Erde grafisch darzustellen, gibt es viele, aber wer will sich schon die Arbeit machen, alle Koordinaten mühsam in den Computer einzutippen? Hier hilft ein Blick in die Public-Domain-Szene. Dort gibt es nämlich oft jemanden, der irgend etwas irgendwann schon einmal gemacht hat. Und so verwundert es auch nicht, daß man in der PC-SIG-Bibliothek drei prall gefüllte Disketten mit den Koordinaten der Welt findet (Nummer 494 bis 496). Leider hat die Sache aber einen Haken. Der neugierige Benutzer wird nämlich vergebens nach einem Programm suchen, mit dem er diesen phantastischen Datenstamm auch grafisch aufbereiten kann.

Um die Daten lesbar zu machen, gibt es ein den Koordinaten beigefügtes Programm, das alle Koordinaten in eine Textdatei im ASCII-Code ablegt. Spätestens wenn Sie 200 KByte mühsam umwandeln, erhalten Sie einen ungefähren Eindruck von der riesigen Datenmenge. Sehen Sie sich die Textdateien mit dem Type-Befehl an, so stellen Sie fest, daß die Koordinaten denkbar einfach abgelegt sind. Immer zwei Zahlen pro Zeile: erst Breitengrad, dann Längengrad. Alle Datenpaare bis zur nächsten Leerzeile in der ASCII-Datei bilden einen Linienzug.

Die große Datenfülle bringt längere Verarbeitungszeiten mit sich

Nicht genug mit den langen Wartezeiten bei der Umwandlung in das ASCII-Format – die Dateien sind dadurch etwa doppelt so lang geworden, so folgt, daß Einlesen einen erheblichen Teil der Verarbeitungszeit beansprucht.

Das Programm zur Konvertierung in das ASCII-Format ist den Koordinaten im C-Quellcode beigefügt. Das verwendete Datenformat ist vom Typ »float«, in Turbo Pascal für Arithmetikprozessor-Besitzer auch »single« genannt.

Ohne Arithmetikprozessor kann man den Datentyp »single«, der in den Dateien vorkommt, natürlich nicht verwenden. Wer sich jedoch einmal die Mühe gemacht hat, das ganze Turbo-Pascal-Anleitungsbuch durchzublättern, weiß, daß dort eine genaue Beschreibung aller bekannten Datenformate zu finden ist. Eine entsprechende Prozedur, die eine 4 Byte lange Single-Zahl in eine 6 Byte lange Real-Zahl umwandelt, finden Sie in der Convert-Unterprozedur zu »AnalyseMps« im nachstehenden Listing »world.pas«.

Folgt auf ein Koordinatenpaar ein Zeilenvorschub (hexadezimal 0A, dezimal 10), so finden Sie das nächste Wertepaar gleich im Anschluß. Ein Byte mit dem Wert eins bedeutet hingegen eine Trennung zweier Linienzüge. Der entsprechende Küstenzug ist also hier zu Ende, und die Daten werden an anderer Stelle fortgesetzt. Es ist wichtig, dies auch bei der Zeichnung zu beachten, sonst sind alle unabhängigen Linienzüge durch verwirrende Querstriche miteinander verbunden. Solch ein Heben des Stiftes vor dem Weiterzeichnen wird in den Koordinatenpaaren, die im Array »buffer« zwischengespeichert werden, durch die Koordinaten (1E28,1E28) vermerkt. Die Zahl 1E28 entspricht einer Eins mit 28 Nullen – eine Zahl also, die niemals eine Koordinate der Erde sein kann. Folglich kann man anhand der Analyse des Feldes »buffer« feststellen, wann man beim Zeichnen an ein solches Koordinatenpaar gelangt. Daraufhin wird der Zeichenstift angehoben (»moveto«) und an anderer Stelle weitergezeichnet (»lineto«).

Außerdem enthalten die Dateien noch den Copyright-Vermerk des Autors. Damit Sie diese Bytes nicht versehentlich als Koordinatenpaare interpretieren, müssen Sie immer erst bis zum nächsten Zeilenvorschub (Byte 10) warten, bevor Sie die folgenden Bytes als Koordinaten auswerten.

Mit »world.pas« sind Sie nun in der Lage, unter Turbo Pascal mit einer Grafikkarte die Welt darzustellen. Ob das Programm dabei die ASCII-Dateien verwenden soll oder die kürzeren Originaldateien, bestimmen Sie selbst, indem Sie gegen Programmende beim Aufruf der Prozedur »zeichne« den letzten Parameter auf »true« für ASCII-Dateien beziehungsweise auf »false« für die Originaldateien setzen. Das Programm verfährt so, daß zunächst 5000 Wertepaare in den Speicher geladen werden, bevor mit dem Zeichnen begonnen wird. Sonst ist das Laufwerk beziehungsweise die Festplatte ständig in Betrieb. 5000 Wertepaare sind angesichts der riesigen Datenmengen zwar nicht sehr viel, erlauben aber eine sinnvoll getrennte Handhabung von Einlesen und Zeichnen der Koordinatenpaare.

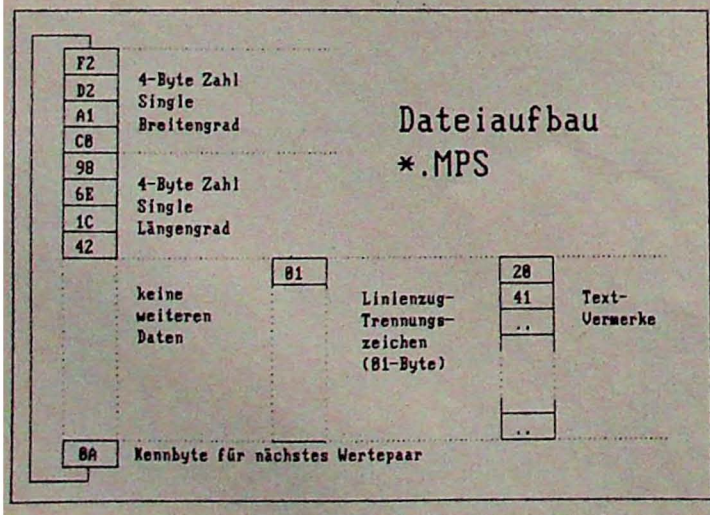


Bild 1. Nach diesem Schema sind die Koordinaten in den Dateien der Public-Domain-Disketten abgelegt

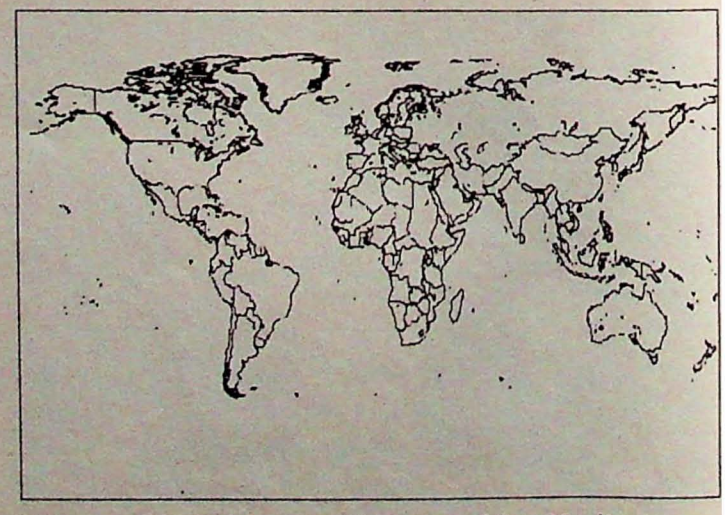


Bild 2. So sieht die komplette Weltkarte auf dem Monitor aus

Nachdem das Programm dann ein knappes Megabyte an Daten umgewälzt hat, sehen Sie auf Ihrem Bildschirm eine komplette Weltkarte inklusive der Landesgrenzen (Bild 2).

Richtig Interessant wird es aber sicherlich erst dann, wenn Sie die Karte für Ihre eigenen Darstellungen verwenden wollen. Dazu belassen Sie die Analyseprozeduren zum Einlesen der Daten so, wie sie sind. Jeweils nach 5000 Wertepaaren oder bei Dateiende wird dann die Prozedur »displaybuffer« aufgerufen, welche die Verarbeitung der Daten übernimmt. Das Programm teilt den Bildschirmbereich in ein Gradsystem auf, wie Sie es vom Atlas her kennen: 180 Breiten- und 360 Längengrade, jeweils unterteilt in nördlich/südlich beziehungsweise westlich/östlich. Nach diesem einfachen System werden die Koordinaten direkt auf die Bildschirmpunkte übertragen.

Um das Programm ausprobieren zu können, müssen Sie nur die Konstanten »pfad« und »graphpath« Ihren eigenen Voraussetzungen anpassen. Unter »pfad« geben Sie Ihren Verzeichnispfad zu den Koordinatendateien an. In weiterführenden Unterpfaden erwartet das Programm dann die Koordinatendateien, wie sie auch auf den PC-SIG-Disketten gespeichert wurden. Die Konstante »graphpath« müssen Sie mit dem Verzeichnispfad zu den Grafiktreibern belegen.

Sinn des Programms ist es, den tieferen Einstieg in die Verarbeitung der Daten vorzubereiten. Die unangenehme Arbeit der Dateianalyse wird Ihnen abgenommen (siehe auch Bild 1), und Sie können sich gleich auf die Gestaltung der Welt auf Ihrem Bildschirm konzentrieren. So gibt es zum Beispiel die verschiedensten Arten der Projektion des Globus auf eine Fläche. Alle Darstellungen haben ihre Vor- und Nachteile. Es gibt winkeltreue Darstellungen, die in der Nautik eine große Rolle spielen, dafür jedoch die Proportionen der Kontinente völlig falsch wiedergeben. Dafür gibt es flächentreue oder abstandstreue Darstellungsmethoden – je nach Einsatzbedarf. Da jedoch alle Projektionen einer Kugel auf eine Ebene Verzerrungen und Untreuen beinhalten, bietet sich besonders die Darstellung als Kugel an. Mit etwas Aufwand bekommt man sogar eine rotierende Weltkugel auf den Bildschirm – eine wahre Augenweide.

Des weiteren besteht die Möglichkeit, statt einer europazentrierten Darstellung eine Karte mit Asien oder Amerika in der Mitte zu zeichnen. Angesichts der hohen Genauigkeit der Wertepaare lohnt es sich auch, einzelne Kontinente vergrößert und zentral darzustellen.

Da ebenfalls die Landesgrenzen digitalisiert werden, ist der Aufruf einzelner Staaten denkbar. Wenn man nun noch eine Datei mit den Hauptstädten und deren Koordinaten anlegt, bekommt man ein sehr nützliches Werkzeug in die Hand.

Allerdings sind hierzu die vorhandenen Daten nicht besonders gut strukturiert. Um nicht ewig auf eine kleine Detailzeichnung warten zu müssen, sollten Sie sich Gedanken über die Umstrukturierung der Daten machen. Sinnvolle Forderungen sind einmal die Möglichkeit zum freien Zoomen beliebiger Ausschnitte und zum anderen die Darstellung beliebiger Staaten. Für einen effizienten Zoom-Algorithmus müssen Sie gezielt auf Daten bestimmter Planquadrate zugreifen können. Eine Umstrukturierung bringt hier wesentliche Zeitvorteile. Wenn die Daten nach Längen- und Breitengraden sortiert abgelegt werden, spart man sich beispielsweise die Zeichnung von ganz Afrika, wenn nur das Mittelmeergebiet interessant ist.

Bei der Staatendarstellung bietet sich die Erfassung von Linienzügen nach Grenzen an. Eine Codeziffer bezeichnet die Länder, die diese Grenze gemeinsam haben. In einer Art Katalog muß dann nur durchgesehen werden, in welchen Grenzzügen die Codenummer für ein bestimmtes Land vorkommt. Nur diese Linien muß das Programm dann zeichnen.

Unerläßlich bei einer Umstrukturierung der riesigen Datenmengen ist ein handlicher Editor, der den Streckenzug und die Koordinaten gleichzeitig anzeigen kann.

Bei Interesse stellen wir Ihnen in einer der nächsten Ausgaben die eine oder andere Beispiellösung zu den oben erwähnten Anwendungen vor.

(Frank Höppner/ue)

```
[K37]    1: { ** Weltdarstellung unter Verwendung ** }
[8S6]    2: { ** des Public-Domain Datenstammes   ** }
[556]    3: { ** TWD (The World Digitized)        ** }
[XN5]    4: { ** von Frank Hoeppner,              ** }
[VZ5]    5: { ** (c) DOS International 1989       ** }
[ØØØ]    6:
[CQ2]    7: USES Graph;
[ØØØ]    8:
[7D6]    9: CONST  Names : ARRAY[1..3Ø] OF STRING[2Ø]
[VB5]   1Ø:        = ('NORTHAME\NAØ', 'NORTHAME\NA1',
[RH2]   11:           'NORTHAME\NA2',
[F85]   12:           'NORTHAME\NA3', 'NORTHAME\USAØ',
[TN2]   13:           'NORTHAME\USA1',
[K35]   14:           'NORTHAME\GRØ', 'NORTHAME\GR1',
[XH2]   15:           'NORTHAME\PA1',
[M65]   16:           'SOUTHAME\SAØ', 'SOUTHAME\SA1',
[W65]   17:           'SOUTHAME\SA2', 'SOUTHAME\SA3',
[1X3]   18:           'EUROPE\EØ', 'EUROPE\E1',
[FX3]   19:           'EUROPE\E2', 'EUROPE\E3',
[P24]   2Ø:           'AFRICA\AFØ', 'AFRICA\AF1',
[234]   21:           'AFRICA\AF2', 'AFRICA\AF3',
[KK3]   22:           'ASIA\ASØ', 'ASIA\AS1',
[UK3]   23:           'ASIA\AS2', 'ASIA\AS3',
[P35]   24:           'AUSTRALI\AUØ', 'AUSTRALI\AU1',
[EH2]   25:           'AUSTRALI\AU2',
[N35]   26:           'ANTARCTI\ANØ', 'ANTARCTI\AN1' );
[HS5]   27:   Pfad       = 'E:\SOURCE\DATA\WORLD\';
[1C4]   28:   GraphPath  = 'D:\TP5\BGI\';
[ØØØ]   29:
[VD6]   3Ø: VAR Buffer    : ARRAY[Ø..4999,Ø..1] OF REAL;
[KW1]   31:     UrSprungX,
[MW1]   32:     UrSprungY,
[W62]   33:     FileCounter,
[EE1]   34:     BufSize,
[M52]   35:     GraphDriver,
[FX3]   36:     GraphMode        : INTEGER;
[ØØØ]   37:
```

```
[VM1]   38: PROCEDURE
[NZ5]   39: DisplayBuffer(endpointer:INTEGER);
[LX6]   4Ø:   VAR bufpointer,x,y,lastx,lasty : INTEGER;
[ØØØ]   41:
[VM1]   42:   PROCEDURE
[776]   43:   CalculateScreenPosition(i:INTEGER);
[MUØ]   44:     BEGIN
[2S5]   45:       x:=Round(UrSprungX+UrSprungX/18Ø*
[M82]   46:                Buffer[i,1]);
[FP5]   47:       y:=Round(UrSprungY-UrSprungY/89*
[H82]   48:                Buffer[i,Ø]);
[2NØ]   49:     END;
[ØØØ]   5Ø:
[MUØ]   51:   BEGIN
[6D2]   52:     bufpointer:=Ø;
[P31]   53:     REPEAT
[3F5]   54:       IF (Buffer[bufpointer,Ø]=1E28) AND
[9H4]   55:          (Buffer[bufpointer,1]=1E28)
[2RØ]   56:          THEN
[MUØ]   57:            BEGIN
[DX3]   58:              REPEAT Inc(bufpointer);
[5YØ]   59:              UNTIL
[YN4]   6Ø:              (Buffer[bufpointer,Ø]<>1E28)
[WLØ]   61:              AND
[NT4]   62:              (Buffer[bufpointer,1]<>1E28);
[A94]   63:              CalculateScreenPosition
[D82]   64:              (bufpointer);
[LZ1]   65:              MoveTo(x,y);
[XC3]   66:              lastx:=x; lasty:=y;
[4MØ]   67:            END
[ZRØ]   68:          ELSE
[MUØ]   69:            BEGIN
[A94]   7Ø:              CalculateScreenPosition
[D82]   71:              (bufpointer);
[V15]   72:            IF NOT((lastx=x) AND (lasty=y))
[TW2]   73:              THEN LineTo(x,y);
```

Listing. »world.pas« wertet den Public-Domain-Datenstamm aus

```
[XC3]   74:              lastx:=x; lasty:=y;
[ZNØ]   75:            END;
[ZR2]   76:        Inc(bufpointer);
[745]   77:      UNTIL bufpointer>=endpointer;
[ZNØ]   78:    END;
[ØØØ]   79:
[6Z4]   8Ø: PROCEDURE LiesFile(a:STRING);
[984]   81:   VAR WorldDataFile : TEXT;
[WT2]   82:       b             : STRING;
[SS3]   83:       x,y,error     : INTEGER;
[MUØ]   84:   BEGIN
[ZH1]   85:     a:=a+'.MP1';
[T14]   86:     ASSIGN(WorldDataFile,a);
[WM3]   87:     RESET(WorldDataFile);
[HL5]   88:     fillchar(buffer,SizeOf(buffer),Ø);
[WZ4]   89:     BufSize:=1; buffer[Ø,Ø]:=1E28;
[AØ3]   9Ø:                 buffer[Ø,1]:=1E28;
[P31]   91:     REPEAT
[HZ3]   92:       READLN(WorldDataFile,b);
[GM2]   93:       IF b='' THEN BEGIN
[573]   94:                     IF bufsize=4999 THEN
[MUØ]   95:                       BEGIN
[U73]   96:                       DisplayBuffer(4999);
[KY1]   97:                         bufsize:=Ø;
[PS2]   98:                         fillchar(buffer,
[VU2]   99:                         SizeOf(buffer),Ø);
[ZNØ]  1ØØ:                       END;
[X13]  1Ø1:                     buffer[bufsize,Ø]
[C11]  1Ø2:                     :=1E28;
[Y13]  1Ø3:                     buffer[bufsize,1]
[ØUØ]  1Ø4:                     :=1E28
[4MØ]  1Ø5:                   END
[WP1]  1Ø6:                 ELSE BEGIN
[N43]  1Ø7:                       IF bufsize=5ØØØ THEN
[MUØ]  1Ø8:                         BEGIN
[XB2]  1Ø9:                           DisplayBuffer
[6XØ]  11Ø:                           (5ØØØ);
[KY1]  111:                           bufsize:=Ø;
[PS2]  112:                           fillchar(buffer,
[VU2]  113:                           SizeOf(buffer),Ø);
[ZNØ]  114:                         END;
[SRØ]  115:                       x:=Ø;
[LN2]  116:                       REPEAT x:=Succ(x)
[VU2]  117:                       UNTIL NOT(b[x] IN
[SA2]  118:                         ['.','Ø'..'9','-']);
[2TØ]  119:                       y:=x;
[NN2]  12Ø:                       REPEAT y:=Succ(y)
[9G3]  121:                       UNTIL (y>LENGTH(b)) OR
[A52]  122:                       NOT(b[y] IN
[SA2]  123:                         ['.','Ø'..'9','-']);
[5R2]  124:                       VAL(COPY(b,1,x-1),
[163]  125:                           buffer[BufSize,Ø],
[291]  126:                           error);
[Ø83]  127:                       VAL(COPY(b,x+1,y-x-1),
[263]  128:                           buffer[BufSize,1],
[291]  129:                           error);
[ZNØ]  13Ø:                     END;
[TD3]  131:       BufSize:=BufSize+1;
[ØØØ]  132:
[484]  133:     UNTIL EOF(WorldDataFile);
[624]  134:     DisplayBuffer(bufsize);
[LL3]  135:     CLOSE(WorldDataFile);
[ZNØ]  136:   END;
[ØØØ]  137:
[1E5]  138: PROCEDURE AnalyseMPS(a:STRING);
[BH2]  139:   VAR f        : FILE;
[ØA3]  14Ø:       fdmy     : FILE OF BYTE;
[BG4]  141:       b        : ARRAY[1..1Ø24] OF BYTE;
[MF2]  142:       fs       : LONGINT;
[XL2]  143:       bp,fp    : INTEGER;
[UK4]  144:       koord    : ARRAY[1..8] OF BYTE;
[SK2]  145:       ende     : BOOLEAN;
[UL2]  146:       gelesen  : WORD;
[ØØØ]  147:
[865]  148:   PROCEDURE ConvertSingleToReal
[SK3]  149:       ( VAR w : REAL; VAR s );
[FN4]  15Ø:     VAR sng : ARRAY[1..4] OF BYTE
[M12]  151:               ABSOLUTE s;
[5X3]  152:           rea : ARRAY[1..6] OF BYTE
[Ø22]  153:               ABSOLUTE w;
[MUØ]  154:     BEGIN
[2R2]  155:       rea[4]:=sng[1];
[6R2]  156:       rea[5]:=sng[2];
[LH4]  157:       rea[6]:=(sng[3] AND 127) OR
[1P2]  158:               (sng[4] AND 128);
[FC5]  159:       rea[1]:=((sng[4] AND 127) SHL 1) OR
[3X3]  16Ø:               ((sng[3] AND 128) SHR 7)+2;
[ZNØ]  161:     END;
[ØØØ]  162:
[MUØ]  163:   BEGIN
[5N1]  164:     a:=a+'.MPS';
[UL4]  165:     ASSIGN(fdmy,a); RESET(fdmy);
[7A5]  166:     fs:=FileSize(fdmy); CLOSE(fdmy);
[8V1]  167:     ASSIGN(f,a);
[VE1]  168:     RESET(f);
[KXØ]  169:     fp:=Ø;
[ØØØ]  17Ø:
```

```
[HL5]  171:       fillchar(buffer,SizeOf(buffer),Ø);
[D12]  172:       ende:=FALSE;
[WZ4]  173:       BufSize:=1; buffer[Ø,Ø]:=1E28;
[AØ3]  174:                   buffer[Ø,1]:=1E28;
[P31]  175:       REPEAT
[DU3]  176:         fillchar(b,SizeOf(b),26);
[Y14]  177:         BlockRead(f,b,8,gelesen);
[AXØ]  178:         bp:=Ø;
[P31]  179:         REPEAT
[242]  18Ø:           bp:=Succ(bp);
[P42]  181:           fp:=Succ(fp);
[552]  182:           fs:=Pred(fs);
[WN1]  183:           CASE fp OF
[1W3]  184:            1..7 : koord[fp]:=b[bp];
[HF1]  185:            8    : BEGIN
[N43]  186:                     IF bufsize=5ØØØ THEN
[MUØ]  187:                       BEGIN
[XB2]  188:                         DisplayBuffer
[6XØ]  189:                         (5ØØØ);
[KY1]  19Ø:                         bufsize:=Ø;
[PS2]  191:                         fillchar(buffer,
[VU2]  192:                         SizeOf(buffer),Ø);
[ZNØ]  193:                       END;
[Z13]  194:                     koord[fp]:=b[bp];
[TM3]  195:                     ConvertSingleToReal
[3D3]  196:                       (buffer[bufsize,Ø],
[EP1]  197:                        koord[1]);
[TM3]  198:                     ConvertSingleToReal
[5D3]  199:                       (buffer[bufsize,1],
[YP1]  2ØØ:                        koord[5]);
[N14]  2Ø1:                     bufsize:=Succ(bufsize);
[ZNØ]  2Ø2:                   END;
[FM4]  2Ø3:           ELSE CASE b[bp] OF   { fp>8 }
[LF1]  2Ø4:                  $A :  fp:=Ø;
[3H1]  2Ø5:                  $1 :  BEGIN
[3F2]  2Ø6:                          IF bufsize=4999
[2RØ]  2Ø7:                          THEN
[MUØ]  2Ø8:                            BEGIN
[XB2]  2Ø9:                              DisplayBuffer
[2ZØ]  21Ø:                              (4999);
[KY1]  211:                              bufsize:=Ø;
[RC1]  212:                              fillchar
[XG2]  213:                              (buffer,SizeOf
[3U1]  214:                               (buffer),Ø);
[ZNØ]  215:                            END;
[X13]  216:                          buffer[bufsize,Ø]
[C11]  217:                          :=1E28;
[Y13]  218:                          buffer[bufsize,1]
[C11]  219:                          :=1E28;
[2D1]  22Ø:                          bufsize
[GL2]  221:                          :=Succ(bufsize);
[ZNØ]  222:                        END;
[ZNØ]  223:                END;
[ZNØ]  224:           END;
[NU1]  225:           ende:=fs=Ø;
[YS3]  226:         UNTIL (bp=1Ø24) OR (ende);
[AY1]  227:       UNTIL ende;
[624]  228:       DisplayBuffer(bufsize);
[ØØØ]  229:
[KD1]  23Ø:     CLOSE(f);
[ZNØ]  231:   END;
[ØØØ]  232:
[VZ2]  233: PROCEDURE Zeichne
[HM4]  234: (a:STRING; decoded : BOOLEAN);
[MUØ]  235:   BEGIN
[9A4]  236:     IF decoded THEN LiesFile(a)
[PC3]  237:                ELSE AnalyseMPS(a);
[ZNØ]  238:   END;
[ØØØ]  239:
[MUØ]  24Ø: BEGIN
[Y13]  241:   WRITELN; WRITELN;
[2P5]  242:   WRITELN('**** The World Digitized'+
[BD5]  243:      'Database Display, PC-SIG #494-#496');
[XF1]  244:   WRITELN;
[UV6]  245:   WRITELN('Darstellung der Welt mit Hilfe'+
[FP5]  246:      'des Public-Domain Datenstammes');
[991]  247:   READLN;
[ØØØ]  248:
[A86]  249:   DetectGraph(GraphDriver,GraphMode);
[WK1]  25Ø:   InitGraph
[EV5]  251:     (GraphDriver,GraphMode,GraphPath);
[MH5]  252:   IF graphresult<>Ø THEN WRITE(#7#7);
[322]  253:   ClearDevice;
[A94]  254:   UrSprungX:=GetMaxX DIV 2;
[F94]  255:   UrSprungY:=GetMaxY DIV 2;
[ØØØ]  256:
[6X3]  257:   FOR FileCounter:=1 TO 3Ø
[L16]  258:       DO Zeichne(pfad+Names[FileCounter],
[361]  259:                  FALSE);
[U23]  26Ø:   WRITE(#7#7#7); READLN;
[522]  261:   CloseGraph;
[KNØ]  262: END.
```

Listing. »world.pas« wertet den Public-Domain-Datenstamm aus

Zeichnen und leben lassen

Wer seine Programme – dem Trend der Zeit folgend – mit Icons (Pictogrammen) aufwerten oder in Spielen bewegte Figuren (Sprites) verwenden möchte, mußte bislang eine erhebliche Zeit für die Realisierung dieser Aufgabe aufwenden. Mit dem Programm »Spred« jedoch erzielen Sie in kurzer Zeit ansprechende Resultate.

Das Turbo-Pascal-Programm stellt Ihnen neun maus- oder tastaturgesteuerte Funktionen zur Verfügung, die Ihnen einfachste Sprite-Erzeugung, -Verwaltung und -Animation gestatten. Sie müssen allerdings vor dem Programmaufruf sicherstellen, daß sich die Datei »cga.bgi« im aktuellen Verzeichnis befindet, oder Sie geben den Pfad als Parameter an.
Nach dem Start können Sie sofort loslegen: Mit dem mausgesteuerten Pfeil setzen Sie im Raster durch Drücken einer Maustaste einen Punkt in der aktuellen Farbe (Bild). Das Ändern der Zeichenfarbe, des Hintergrunds, der Palette oder das Löschen des Sprite geschieht durch Anklicken des jeweiligen Schriftzuges oder durch Betätigen des farblich hervorgehobenen Buchstabens.
Sie werden bemerken, daß einige Menüpunkte doppelt vorkommen – das hat seinen Sinn. Über dem ersten Optionenblock steht »Sprite«; dadurch wird angedeutet, daß sich die nachfolgenden Punkte nur auf das einzelne Sprite beziehen, das sich rechts daneben im Raster befindet. Es wird streng zwischen dem »Sprite« und der »Sequenz« unterschieden. Auf diese »Sequenz« bezieht sich nämlich der nächste Optionenblock.
Machen Sie sich nun mit den einzelnen Menüpunkten vertraut. Die ersten drei Menüs unter dem Optionenblock »Sprite« funktionieren dabei analog zu den gleichnamigen Punkten des Optionenblocks »Sequenz«.
Wenn Sie den Punkt »Laden« wählen, so erscheint die Abfrage nach dem Dateinamen, den Sie dann ohne Dateierweiterung eingeben müssen. Sie haben hier – wie übrigens an jeder Stelle des Programms – die Möglichkeit, durch Drücken der Esc-Taste die laufende Aktion abzubrechen und zum Hauptmenü zurückzukehren. Existiert die gewünschte Datei, so wird sie geladen und das Sprite in das Gitter eingetragen. Befindet sich bereits ein Sprite im Raster, so wird es überschrieben. Das Speichern geht genauso einfach vonstatten.
Wenn Sie die Sprites auf eine Diskette oder Festplatte speichern, so handelt es sich dabei immer um den Inhalt des Arrays, in dem das Programm das Bild ablegt. Sie könnten diese Datei von Ihren eigenen Programmen wieder einlesen und weiterbearbeiten lassen; dies ist jedoch mit erheblichem Zeitaufwand verbunden. Der Menüpunkt »Quellcode« erzeugt nun eine Textdatei in Form einer Unit. In dieser existiert nur ein Const-Deklarationsteil, der das Bild mit dem Namen »Sprite« initialisiert. Diesen Namen dürfen Sie beliebig ändern. Nach dem Compilieren der Unit läßt sie sich in eigene Programme einbinden. Die Sprites stehen Ihnen dann unter dem gewählten Namen ohne Verzögerung zur Verfügung.
Klicken Sie »Quellcode« an, und geben Sie den Dateinamen ein. Es folgt die Aufforderung, einen Ausschnitt aus dem rechtsstehenden Gitter zu markieren. Benutzen Sie die Maus, um die linke obere und die rechte untere Ecke des Rechtecks festzulegen, von dem der Quellcode erzeugt wird. Diese Funktion wurde implementiert, da Sie häufiger Sprites mit kleineren Ausmaßen als 40x40 Bildpunkte benötigen. Möchten Sie die erste Markierung zurücknehmen, so wählen Sie sie einfach noch einmal an.
Machen Sie nun einen Sprung zum vierten Menüpunkt (»Manipulieren«) des Optionenblocks »Sequenz«. Hier wird Ihnen gleich deutlich, was sich hinter einer Sequenz verbirgt. Sie sehen nun zwölf Kästchen in drei Reihen zu je vier Stück sowie ein einzelnes Kästchen rechts oben, welches das Sprite aus dem Raster im Hauptmenü zeigt. Eine Sequenz besteht somit aus zwölf einzelnen Sprites, wobei jedes mit einem Buchstaben gekennzeichnet ist. Mit Hilfe dieser Sequenz können Sie zusammengehörende Bilder gesammelt erfassen und animieren. Es stehen Ihnen dazu drei Modi zur Verfügung.
Die Modi »Vertauschen« und »Kopieren« erfordern die Markierung zweier Sprites, beim Kopieren müssen Sie jedoch beachten, daß die Reihenfolge nicht gleichgültig ist. Wollen Sie zwei Sprites vertauschen, so wählen Sie diese Option und klicken dann hintereinander zwei Bilder an. Um die erste Markierung zurückzunehmen, genügt wieder ein erneutes Anwählen dieses Sprite. Beim Kopieren sollten Sie Vorsicht walten lassen: Zuerst muß das zu kopierende Sprite gewählt werden. Dann folgt das Ziel. Das dort liegende Sprite wird beim Kopiervorgang gelöscht.
Zum Testen, ob die Sequenz bei der Animation den gewünschten Effekt erzielt, stellt Ihnen die Option »Animieren« im Hauptmenü eine nützliche Routine zur Verfügung. In dem großen Kasten wird nun Sprite A gezeigt und kurz darauf gelöscht. Es folgt Sprite B und so weiter. Jedes Bild wird dabei immer etwas verschoben, wodurch eine Bewegung simuliert wird. Welche Bilder der Sequenz verwendet werden, legen Sie unter dem Punkt »Sprites« fest. Ist das letzte erreicht, so wird wieder von vorne begonnen.
Der letzte Punkt (»Geschwindigkeit«) bestimmt die Zeitspanne, die vergeht, bis ein Bild gelöscht und das nächste dargestellt wird. Je größer die Zahl, um so schneller erfolgt der Wechsel. Sie verlassen das Programm, indem Sie den Menüpunkt »Beenden« wählen.

(Robert S. Krieger/zi)

Zeichnen Sie Ihre Sprites mit Hilfe der Maus

Name:	spred.pas
Sprache	Turbo Pascal 4.0 und 5.0
Grafikkarte	CGA
Besonderheiten	keine

```
[ZC5]    1: { Programmname: SPRITEED.PAS }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[ZE4]    5: program SpriteGenerator;
[ØØØ]    6:
[9A4]    7: uses (* SpriteGenerator *)
[MW3]    8:   Crt, Dos, Graph, Graph3;
[ØØØ]    9:
[6XØ]   1Ø: const
[CG2]   11:   Nix : Char = #222;
[ØØØ]   12:
[STØ]   13: type
[YV3]   14:   String7Ø = String [7Ø];
[FR5]   15:   SpriteTyp = Array [1..4Ø4] of Byte;
[ØØØ]   16:
[RNØ]   17: var
[V96]   18:   Sprite     :  Array [1..14] of SpriteTyp;
[D52]   19:   BeginSprite,
[DM4]   2Ø:   EndSprite          :   String[1];
[2Z1]   21:   XPos, YPos,
[1G4]   22:   V, StepX, StepY    : Integer;
[N52]   23:   Hintergrund,
[554]   24:   PalettenNr, Farbe : Byte;
[ØØØ]   25:
[7R2]   26: procedure Init;
[7T2]   27:  var SZ : Integer;
[L72]   28: begin (* Init *)
[CH2]   29:   PalettenNr :=  2;
[UR2]   3Ø:   Hintergrund := Ø;
[DN1]   31:   Farbe := 3;
[GZØ]   32:   V := 1;
[UR1]   33:   StepX := 1;
[TR1]   34:   StepY := Ø;
[P23]   35:   BeginSprite := 'A';
[MT2]   36:   EndSprite :=   'L';
[X62]   37: end (* Init *);
[ØØØ]   38:
[F36]   39: procedure Box (X1, Y1, X2, Y2, F: Integer);
[MUØ]   4Ø: begin
[742]   41:   SetColor (F);
[PB2]   42:   MoveTo (X1,Y1);
[5A2]   43:   LineTo (X2,Y1);
[7A2]   44:   LineTo (X2,Y2);
[4A2]   45:   LineTo (X1,Y2);
[P22]   46:   LineTo (X1,Y1)
[7W1]   47: end (* Box *);
[ØØØ]   48:
[WC4]   49: procedure FillBox (X1, Y1, X2,
[1L2]   5Ø:                    Y2, F : Integer);
[TS2]   51:  var LZ : Integer;
[MU2]   52: begin (* FillBox *)
[742]   53:   SetColor (F);
[A96]   54:   for LZ := Y1 to Y2 do Line (X1,LZ,X2,LZ);
[GP2]   55:   SetColor (Farbe)
[LL2]   56: end (* FillBox *);
[ØØØ]   57:
[WL5]   58: procedure WA (X,Y,Farbe : Integer;
[4C3]   59:                TextStr : String7Ø);
[MUØ]   6Ø: begin
[6U2]   61:   SetColor (Farbe);
[334]   62:   OutTextXY (X,Y,TextStr)
[KL1]   63: end (* WA *);
[ØØØ]   64:
[MG5]   65: function LiesT (X1,X2,Y : Integer;
[PR2]   66:                 Art, AlterWert :
[GC3]   67:                      String7Ø) : String7Ø;
[Ø43]   68:  var HS :    String7Ø;
[8U1]   69:      SZ :    Byte;
[712]   7Ø:      Taste : Char;
[9D2]   71: begin (* LiesT *)
[P21]   72:   HS := '';
[K63]   73:   if Art <> 'Datei' then
[HR3]   74:     FillBox (X1,Y,X2,Y+7,Ø);
[P31]   75:   repeat
[HR2]   76:     Taste := ReadKey;
[LK5]   77:     if Taste = #Ø then Taste := ReadKey;
[V15]   78:     if Length (HS) < (X2-X1) div 8 then
[353]   79:       if (Art = 'AStep') and
[SP3]   8Ø:          (Taste in ['-','Ø'..'3'])  or
[MV2]   81:          (Art = 'Datei') and
[T74]   82:          (Taste in ['Ø'..'9','a'..'z','!',
[5K2]   83:                     '#'..')','@','^','_',
[NG2]   84:                     '{','}','~']) or
[F53]   85:          (Art = 'ASprite') and
[UD3]   86:          (Taste in ['a'..'l']) or
[N82]   87:          (Art = 'AV') and
[XP3]   88:          (Taste in ['Ø'..'9']) then
[MUØ]   89:       begin
[4F3]   9Ø:         WA (Length(HS)*8+X1,Y,3,
[LF2]   91:             UpCase (Taste));
[6H3]   92:         HS := HS + UpCase (Taste)
[ZNØ]   93:       end;
[YN2]   94:     if Taste = #8 then
[MUØ]   95:     begin
[8P4]   96:       HS := Copy (HS,1,Length(HS)-1);
[8H4]   97:       FillBox (Length(HS)*8+X1,Y,
[SF3]   98:                Length(HS)*8+X1+7,Y+7,Ø)
[4MØ]   99:     end
[ZT3]  1ØØ:   until Taste in [#13,#27];
[ES4]  1Ø1:   if Taste = #13 then LiesT := HS;
[UV4]  1Ø2:   if (Taste = #27) or (HS = '') then
[MUØ]  1Ø3:   begin
[V93]  1Ø4:     LiesT := AlterWert;
[K63]  1Ø5:     if Art <> 'Datei' then
[HR3]  1Ø6:       FillBox (X1,Y,X2,Y+7,Ø);
[C53]  1Ø7:     WA (X1,Y,3,AlterWert)
[4MØ]  1Ø8:   end
[W62]  1Ø9: end (* LiesT *);
[ØØØ]  11Ø:
[2F3]  111: procedure MouseEin;
[RE3]  112:  var Regs : Registers;
[NY2]  113: begin (* MouseEin *)
[LY1]  114:   Regs.AX := 1;
[V92]  115:   Intr ($33,Regs)
[5Z2]  116: end (* MouseEin *);
[ØØØ]  117:
[5G3]  118: procedure MouseAus;
[RE3]  119:  var Regs : Registers;
[VY2]  12Ø: begin (* MouseAus *)
[MY1]  121:   Regs.AX := 2;
[V92]  122:   Intr ($33,Regs)
[NØ3]  123: end (* MouseAus *);
[ØØØ]  124:
[GL3]  125: procedure InitMouse;
[RE3]  126:  var Regs : Registers;
[B53]  127: begin (* InitMouse *)
[KY1]  128:   Regs.AX := Ø;
[PB2]  129:   Intr ($33,Regs);
[TF1]  13Ø:   MouseEin
[UØ3]  131: end (* InitMouse *);
[ØØØ]  132:
[ØY3]  133: {*** M A I N   S C R E E N S ***}
[ØØØ]  134:
[6P3]  135: procedure MainScreen;
[3G5]  136:  var SZ, Treiber, Modus : Integer;
[5B3]  137: begin (* MainScreen *)
[1F2]  138:   Treiber := CGA;
[ZB2]  139:   Modus := CGACØ;
[WF3]  14Ø:   if ParamCount = 1 then
[TØ6]  141:     InitGraph (Treiber,Modus,ParamStr(1))
[ZRØ]  142:   else
[ZG4]  143:     InitGraph (Treiber,Modus,'');
[WS3]  144:   if GraphResult <> Ø then
[MUØ]  145:   begin
[N53]  146:     TextBackGround (Ø);
[3E2]  147:     TextColor (15);
[DB1]  148:     ClrScr;
[6H5]  149:     WriteLn (#7,'Achtung ! BGI Treiber '+
[233]  15Ø:              'nicht gefunden.');
[RPØ]  151:     halt
[ZNØ]  152:   end;
[4H3]  153:   DirectVideo := False;
[Z73]  154:   CheckBreak  := False;
[UV2]  155:   for SZ := 1 to 14 do
[925]  156:     GetImage (Ø,Ø,39,39,Sprite[SZ]);
[4L2]  157:   Box (Ø,Ø,319,28,2);
[KH2]  158:   Line (Ø,14,319,14);
[8N4]  159:   WA (2Ø, 4,3,'SpriteEditor 1.Ø   '+
[DW2]  16Ø:                'by R. S. Krieger');
[NP2]  161:   WA (19, 3,2,'S     E');
[ØH4]  162:   OutTextXY (56,18,'(c) 1989 DOS '+
[WU2]  163:              'International');
[723]  164:   WA (55,17,3,'         DOS '+
[WU2]  165:              'International');
[W85]  166:   SetViewPort (Ø,3Ø,319,199,ClipOff)
[Z13]  167: end (* MainScreen *);
[ØØØ]  168:
[PY4]  169: {*** S P R I T E   E D I T I E R E N ***}
[ØØØ]  17Ø:
[W46]  171: procedure SpriteInGitterEintragen;
[4W3]  172:  var SZ, SZ2, SZ3 : Byte;
[UU5]  173: begin (* SpriteInGitterEintragen *)
[NV2]  174:   for SZ := 1 to 4Ø do
[B23]  175:     for SZ2 := 1 to 4Ø do
[MUØ]  176:     begin
[KH5]  177:       SetColor (GetPixel (1Ø2+SZ2,5+SZ));
[UØ3]  178:       for SZ3 := 1 to 3 do
[1W3]  179:         Line (153+SZ2*4,1+SZ*4+SZ3,
[473]  18Ø:               153+SZ2*4+2,1+SZ*4+SZ3)
[ZNØ]  181:     end;
[GP2]  182:   SetColor (Farbe)
[9K5]  183: end (* SpriteInGitterEintragen *);
[ØØØ]  184:
[W78]  185: procedure EditierenScreen (CVP, Gitter : Bo
            olean);
```

```
[733]  186:  var SZ :      Integer;
[7N2]  187:      Gleich : Boolean;
[953]  188:      H,P,F :  String[2];
[984]  189: begin (* EditierenScreen *)
[SM1]  190:   MouseAus;
[1D4]  191:   if CVP then ClearViewPort;
[D43]  192:   WA (3, 5,3,'Sprite :');
[S24]  193:   OutTextXY (6,16,'- Laden');
[XV4]  194:   OutTextXY (6,26,'- Speichern');
[GV4]  195:   OutTextXY (6,36,'- QuellCode');
[WH4]  196:   OutTextXY (3,47,'Sequenz :');
[G34]  197:   OutTextXY (6,59,'- Laden');
[NW4]  198:   OutTextXY (6,69,'- Speichern');
[8W4]  199:   OutTextXY (6,79,'- QuellCode');
[1E5]  200:   OutTextXY (6,89,'- Manipulieren');
[RW4]  201:   OutTextXY (6,99,'- Animieren');
[000]  202:
[PD3]  203:   Str (PalettenNr,P);
[SE3]  204:   Str (Hintergrund,H);
[E62]  205:   Str (Farbe,F);
[000]  206:
[SB5]  207:   OutTextXY (3,110,'Palette :      '+P);
[VS5]  208:   OutTextXY (3,120,'Hintergrund : '+H);
[E03]  209:   WA (3,130,3,'Farbe :');
[LW2]  210:   WA (115,130,Farbe,F);
[EA3]  211:   WA (3,140,3,'Loeschen');
[294]  212:   OutTextXY (3,150,'Beenden');
[022]  213:   WA (21,15,2,'L');
[5C3]  214:   OutTextXY (21,25,'S');
[1C3]  215:   OutTextXY (21,35,'Q');
[5B3]  216:   OutTextXY (37,58,'d');
[DB3]  217:   OutTextXY (37,68,'e');
[MD3]  218:   OutTextXY (29,78,'u');
[8C3]  219:   OutTextXY (21,88,'M');
[WA3]  220:   OutTextXY (21,98,'A');
[UB3]  221:   OutTextXY (2,109,'P');
[YA3]  222:   OutTextXY (2,119,'H');
[UA3]  223:   OutTextXY (2,129,'F');
[FE3]  224:   OutTextXY (10,139,'o');
[KA3]  225:   OutTextXY (2,149,'B');
[000]  226:
[YP2]  227:   Box (102,5,143,46,3);
[HV2]  228:   for SZ := 0 to 40 do
[UB3]  229:     if Frac (SZ/5) = 0 then
[MU0]  230:     begin
[V12]  231:       SetColor (3);
[9T3]  232:       Line (153,5+SZ*4,319,5+SZ*4)
[4M0]  233:     end
[ZR0]  234:     else
[MU0]  235:     begin
[L12]  236:       SetColor (1);
[CT3]  237:       Line (156,5+SZ*4,316,5+SZ*4)
[ZN0]  238:     end;
[000]  239:
[HV2]  240:   for SZ := 0 to 40 do
[UB3]  241:     if Frac (SZ/5) = 0 then
[MU0]  242:     begin
[V12]  243:       SetColor (3);
[D54]  244:       Line (156+SZ*4,2,156+SZ*4,168)
[4M0]  245:     end
[ZR0]  246:     else
[MU0]  247:     begin
[L12]  248:       SetColor (1);
[A54]  249:       Line (156+SZ*4,5,156+SZ*4,165)
[ZN0]  250:     end;
[BT4]  251:   PutImage (103,6,Sprite[13],0);
[6U2]  252:   SetColor (Farbe);
[8E2]  253:   Gleich := True;
[4Y2]  254:   for SZ := 5 to 404 do
[CF6]  255:     if Sprite[13,SZ] <> Sprite[14,SZ] then
[UF2]  256:       Gleich := False;
[TL4]  257:   if Gitter and not Gleich then
[BF4]  258:     SpriteInGitterEintragen;
[TF1]  259:   MouseEin
[K14]  260: end (* EditierenScreen *);
[000]  261:
[MS5]  262: { S E Q U E N Z  M A N I P U L I E R E N }
[000]  263:
[6B4]  264: procedure SequenzScreen;
[ZT4]  265:  var HelpSprite :   SpriteTyp;
[0C3]  266:      Regs :         Registers;
[C91]  267:      SZ, SZ2,
[GR2]  268:      Marke1X, Marke1Y,
[KZ2]  269:      Marke1XY, Marke2X,
[P24]  270:      Marke2Y, Marke2XY : Byte;
[CG3]  271:      Modus, Taste      : Char;
[LW3]  272: begin (* SequenzScreen *)
[SM1]  273:   MouseAus;
[B85]  274:   GetImage (103,6,142,45,Sprite[13]);
[LK2]  275:   ClearViewPort;
[1Y1]  276:   Modus := 'V';
[113]  277:   WA (7,2,3,'Sequenz :'+
[VA3]  278:                          Sprite :');
```

```
[SW2]  279:   for SZ := 0 to 2 do
[N03]  280:     for SZ2 := 0 to 3 do
[MU0]  281:     begin
[BF3]  282:       Box (SZ2*53+7,SZ*53+13,
[TZ2]  283:            SZ2*53+48,SZ*53+54,3);
[W24]  284:       PutImage (SZ2*53+8,SZ*53+14,
[DM3]  285:                 Sprite[SZ*4+SZ2+1],0);
[0K4]  286:       OutTextXY (SZ2*53+23,SZ*53+56,
[FR2]  287:                  Chr (SZ*4+SZ2+65))
[ZN0]  288:     end;
[0X2]  289:   Box (231,13,272,54,3);
[EE3]  290:   OutTextXY (250,57,'M');
[BY4]  291:   PutImage (232,14,Sprite[13],0);
[CC2]  292:   WA (222, 88,3,'V');
[9A2]  293:   WA (221,87,2,'V');
[FU3]  294:   WA (222,88,1,' ertauschen');
[EM3]  295:   WA (222,103,3,'Kopieren');
[DC2]  296:   WA (237,102,2,'p');
[RM3]  297:   WA (222,118,3,'Loeschen');
[2D2]  298:   WA (245,117,2,'s');
[FL2]  299:   WA (222,133,3,'ESC');
[5L2]  300:   WA (221,132,2,'ESC');
[FL1]  301:   MouseEin;
[YY1]  302:   Marke1X := 0;
[2Z1]  303:   Marke1Y := 0;
[1Z1]  304:   Marke2X := 0;
[5Z1]  305:   Marke2Y := 0;
[T62]  306:   Marke1XY := 0;
[W62]  307:   Marke2XY := 0;
[000]  308:
[P31]  309:   repeat
[2K1]  310:     XPos := 0;
[4K1]  311:     YPos := 0;
[NY1]  312:     Regs.AX := 3;
[PB2]  313:     Intr ($33,Regs);
[YK3]  314:     if Regs.BX and 3 > 0 then
[MU0]  315:     begin
[LP2]  316:       XPos := Regs.CX;
[SP2]  317:       YPos := Regs.DX;
[5E1]  318:       Delay(200)
[ZN0]  319:     end;
[000]  320:
[P12]  321:     Taste := Nix;
[Y23]  322:     if KeyPressed then
[MU0]  323:     begin
[HR2]  324:       Taste := ReadKey;
[SG5]  325:       if Taste = #0 then Taste := ReadKey
[ZN0]  326:     end;
[000]  327:
[H24]  328:     if (UpCase (Taste) = 'V') or
[864]  329:        (XPos>440) and (XPos<618) and
[BF4]  330:        (YPos>115) and (YPos<124) then
[MU0]  331:     begin
[SM1]  332:       MouseAus;
[1Y1]  333:       Modus := 'V';
[M24]  334:       WA (222, 88,1,' ertauschen');
[YF3]  335:       WA (222,103,3,'Ko ieren');
[9U4]  336:       OutTextXY (222,118,'Loe chen');
[TF1]  337:       MouseEin
[ZN0]  338:     end;
[000]  339:
[T14]  340:     if (UpCase (Taste) = 'P') or
[T64]  341:        (XPos>442) and (XPos<570) and
[CF4]  342:        (YPos>130) and (YPos<139) then
[MU0]  343:     begin
[SM1]  344:       MouseAus;
[9X1]  345:       Modus := 'p';
[UF3]  346:       WA (222,103,1,'Ko ieren');
[S24]  347:       WA (222, 88,3,' ertauschen');
[9U4]  348:       OutTextXY (222,118,'Loe chen');
[TF1]  349:       MouseEin
[ZN0]  350:     end;
[000]  351:
[524]  352:     if (UpCase (Taste) = 'S') or
[T64]  353:        (XPos>442) and (XPos<570) and
[UF4]  354:        (YPos>145) and (YPos<154) then
[MU0]  355:     begin
[SM1]  356:       MouseAus;
[NX1]  357:       Modus := 's';
[7E3]  358:       WA (222,118,1,'Loe chen');
[YF3]  359:       WA (222,103,3,'Ko ieren');
[MC5]  360:       OutTextXY (222, 88,' ertauschen');
[TF1]  361:       MouseEin
[ZN0]  362:     end;
[000]  363:
[LB5]  364:     if UpCase (Taste) in ['A'..'L'] then
[MU0]  365:     begin
[B35]  366:       XPos := ((Ord (UpCase (Taste)) - 65)
[KB2]  367:                mod 4 * 53 + 10) *2;
[D35]  368:       YPos := ((Ord (UpCase (Taste)) - 65)
[KZ1]  369:                div 4 * 53 + 45)
[ZN0]  370:     end;
```

Listing. »spred.pas« macht das Arbeiten mit Sprites einfach

```
[ØØØ]  371:
[Ø44]  372:     if UpCase (Taste) = 'M' then
[MUØ]  373:     begin
[TT1]  374:       XPos := 47Ø;
[DH1]  375:       YPos := 5Ø
[ZNØ]  376:     end;
[ØØØ]  377:
[SW2]  378:     for SZ := Ø to 2 do
[NØ3]  379:       for SZ2 := Ø to 3 do
[494]  38Ø:         if (XPos div 2 > SZ2*53+7) and
[VØ4]  381:            (XPos div 2 < SZ2*53+48) and
[FØ3]  382:            (YPos > SZ*53+43) and
[AA3]  383:            (YPos < SZ*53+84) then
[MUØ]  384:         begin
[453]  385:           if Marke1XY = Ø then
[MUØ]  386:           begin
[SM1]  387:             MouseAus;
[PS3]  388:             Marke1XY := SZ*4+SZ2+1;
[6K2]  389:             Marke1X  := SZ2;
[LH2]  39Ø:             Marke1Y  := SZ;
[R12]  391:             SetColor (2);
[ØK4]  392:             OutTextXY (SZ2*53+23,SZ*53+56,
[VY2]  393:                        Chr (SZ*4+SZ2+65));
[TF1]  394:             MouseEin
[4MØ]  395:           end
[ZRØ]  396:           else
[MUØ]  397:           begin
[TS3]  398:             Marke2XY := SZ*4+SZ2+1;
[9K2]  399:             Marke2X  := SZ2;
[AA2]  4ØØ:             Marke2Y  := SZ
[4MØ]  4Ø1:           end
[ZNØ]  4Ø2:         end;
[ØØØ]  4Ø3:
[MD3]  4Ø4:     if (XPos div 2 > 231) and
[VØ3]  4Ø5:        (XPos div 2 < 273) and
[VS4]  4Ø6:        (YPos > 43) and (YPos < 84) then
[453]  4Ø7:       if Marke1XY = Ø then
[MUØ]  4Ø8:       begin
[SM1]  4Ø9:         MouseAus;
[FD2]  41Ø:         Marke1XY := 13;
[492]  411:         WA (25Ø,57,2,'M');
[TF1]  412:         MouseEin
[4MØ]  413:       end
[KØ3]  414:       else Marke2XY := 13;
[ØØØ]  415:
[PØ4]  416:     if (Modus in ['V','p']) and
[YU2]  417:        (Marke1XY <> Ø) and
[UL2]  418:        (Marke2XY <> Ø) or
[BE2]  419:        (Modus = 's') and
[4Y2]  42Ø:        (Marke1XY <> Ø) then
[MUØ]  421:     begin
[SM1]  422:       MouseAus;
[2P4]  423:       if Marke1XY <> Marke2XY then
[MUØ]  424:       begin
[EY2]  425:         if (Marke1XY = 13) or
[TY2]  426:         (Marke2XY = 13) then
[MUØ]  427:         begin
[P83]  428:          if Marke1XY = 13 then
[Z32]  429:           case Modus of
[3G1]  43Ø:            'V': begin
[HP3]  431:                   PutImage (Marke2X*53+8,
[A32]  432:                             Marke2Y*53+14,
[VC2]  433:                             Sprite[13],Ø);
[ES3]  434:                   PutImage (232,14,Sprite
[L32]  435:                             [Marke2XY],Ø)
[ZNØ]  436:                 end;
[C54]  437:            'p': PutImage (Marke2X*53+8,
[A32]  438:                           Marke2Y*53+14,
[VC2]  439:                           Sprite[13],Ø);
[Y13]  44Ø:            's': PutImage (232,14,
[C72]  441:                           Sprite[14],Ø)
[4MØ]  442:           end
[ZRØ]  443:           else
[Z32]  444:           case Modus of
[3G1]  445:            'V': begin
[GP3]  446:                   PutImage (Marke1X*53+8,
[732]  447:                             Marke1Y*53+14,
[VC2]  448:                             Sprite[13],Ø);
[ES3]  449:                   PutImage (232,14,Sprite
[F32]  45Ø:                             [Marke1XY],Ø)
[ZNØ]  451:                 end;
[574]  452:            'p': PutImage (232,14,Sprite
[F32]  453:                           [Marke1XY],Ø)
[4MØ]  454:           end
[4MØ]  455:         end
[ZRØ]  456:         else
[Z32]  457:           case Modus of
[3G1]  458:            'V': begin
[GP3]  459:                   PutImage (Marke1X*53+8,
[732]  46Ø:                             Marke1Y*53+14,
[Y41]  461:                             Sprite
[492]  462:                             [Marke2XY],Ø);
[HP3]  463:                   PutImage (Marke2X*53+8,
[A32]  464:                             Marke2Y*53+14,
[Y41]  465:                             Sprite
[F32]  466:                             [Marke1XY],Ø)
[ZNØ]  467:                 end;
[C54]  468:            'p': PutImage (Marke2X*53+8,
[A32]  469:                           Marke2Y*53+14,
[Y41]  47Ø:                           Sprite
[Ø92]  471:                           [Marke1XY],Ø);
[L54]  472:            's': PutImage (Marke1X*53+8,
[732]  473:                           Marke1Y*53+14,
[C72]  474:                           Sprite[14],Ø)
[ZNØ]  475:           end;
[ØØØ]  476:
[Z32]  477:         case Modus of
[3G1]  478:          'V': begin
[H52]  479:                 HelpSprite :=
[173]  48Ø:                 Sprite[Marke1XY];
[DK3]  481:                 Sprite[Marke1XY] :=
[373]  482:                 Sprite[Marke2XY];
[FK3]  483:                 Sprite[Marke2XY] :=
[XV1]  484:                 HelpSprite
[ZNØ]  485:               end;
[4Z3]  486:           'p': Sprite[Marke2XY] :=
[173]  487:                Sprite[Marke1XY];
[AZ3]  488:           's': Sprite[Marke1XY] :=
[UX1]  489:                Sprite[14]
[4MØ]  49Ø:          end
[ZNØ]  491:       end;
[P83]  492:       if Marke1XY = 13 then
[M32]  493:         WA (25Ø,57,3,'M')
[ZG3]  494:       else WA (Marke1X *53+23,
[7B2]  495:                Marke1Y*53+56,3,
[CY2]  496:                Chr (Marke1XY+64));
[T62]  497:       Marke1XY := Ø;
[W62]  498:       Marke2XY := Ø;
[TF1]  499:       MouseEin
[4MØ]  5ØØ:     end
[493]  5Ø1:   until (Taste = #27) or
[Y64]  5Ø2:         (XPos>44Ø) and (XPos<488) and
[2U3]  5Ø3:         (YPos>161) and (YPos<169);
[2K1]  5Ø4:   XPos := Ø;
[4K1]  5Ø5:   YPos := Ø;
[PS4]  5Ø6:   EditierenScreen (True,True)
[6S3]  5Ø7: end (* SequenzScreen *);
[ØØØ]  5Ø8:
[E95]  5Ø9: {*  S E Q U E N Z   A N I M I E R E N  *}
[ØØØ]  51Ø:
[HZ4]  511: procedure SequenzAnimation;
[HR3]  512:  var Help :  String[2];
[7X2]  513:      Regs :  Registers;
[W74]  514:      SpriteNr, Check, SpriteX,
[564]  515:      SpriteY, SiCo : Integer;
[7N2]  516:      Taste         : Char;
[2E4]  517: begin (* SequenzAnimation *)
[SM1]  518:   MouseAus;
[B85]  519:   GetImage (1Ø3,6,142,45,Sprite[13]);
[LK2]  52Ø:   ClearViewPort;
[ML2]  521:   Box (Ø,5,18Ø,16Ø,3);
[A64]  522:   WA (19Ø,2Ø,3,'Sprites  (A..L)');
[M54]  523:   OutTextXY (19Ø, 3Ø,' von : '+
[Ø92]  524:              BeginSprite);
[734]  525:   OutTextXY (19Ø, 4Ø,' bis : '+
[PY1]  526:              EndSprite);
[WY2]  527:   OutTextXY (19Ø, 6Ø,
[NS2]  528:              'Richtung (-3..3)');
[NY2]  529:   Str (StepX,Help);
[AV4]  53Ø:   OutTextXY (19Ø, 7Ø,' x : '+ Help);
[SY2]  531:   Str (StepY,Help);
[FV4]  532:   OutTextXY (19Ø, 8Ø,' y : '+ Help);
[456]  533:   OutTextXY (19Ø,1ØØ,'Geschwindigkeit');
[S64]  534:   OutTextXY (19Ø,11Ø,'(Ø..9) : '+
[2P1]  535:              Chr (V+48));
[K34]  536:   OutTextXY (19Ø,13Ø,'ESC ');
[EB2]  537:   WA (197,29,2,'v');
[1F3]  538:   OutTextXY (197,39,'b');
[UF3]  539:   OutTextXY (197,69,'x');
[WF3]  54Ø:   OutTextXY (197,79,'y');
[GF3]  541:   OutTextXY (189,99,'G');
[TZ3]  542:   OutTextXY (189,129,'ESC');
[FL1]  543:   MouseEin;
[S36]  544:   SpriteNr := Ord (BeginSprite[1])-64;
[H92]  545:   SpriteX  := Ø;
[GG2]  546:   SpriteY  := 8Ø;
[P31]  547:   repeat
[2K1]  548:     XPos := Ø;
[4K1]  549:     YPos := Ø;
[NY1]  55Ø:     Regs.AX := 3;
[PB2]  551:     Intr ($33,Regs);
[YK3]  552:     if Regs.BX and 3 > Ø then
[MUØ]  553:     begin
[LP2]  554:       XPos := Regs.CX;
[XM2]  555:       YPos := Regs.DX
[ZNØ]  556:     end;
```

```
[ØØØ]  557:
[P12]  558:      Taste := Nix;
[Y23]  559:      if KeyPressed then
[MUØ]  56Ø:      begin
[HR2]  561:        Taste := ReadKey;
[SG5]  562:        if Taste = #Ø then Taste := ReadKey
[ZNØ]  563:      end;
[ØØØ]  564:
[H24]  565:      if (UpCase (Taste) = 'V') or
[B64]  566:         (XPos>39Ø) and (XPos<444) and
[484]  567:         (YPos>58) and (YPos<66) then
[MUØ]  568:      begin
[SM1]  569:        MouseAus;
[RC2]  57Ø:        WA (2Ø6, 3Ø,1,'on');
[UV4]  571:        BeginSprite := LiesT (246,254,3Ø,
[LM1]  572:                              'ASprite',
[Ø92]  573:                              BeginSprite);
[VC2]  574:        WA (2Ø6, 3Ø,3,'on');
[S36]  575:        SpriteNr := Ord (BeginSprite[1])-64;
[TF1]  576:        MouseEin
[ZNØ]  577:      end;
[ØØØ]  578:
[1Ø4]  579:      if (UpCase (Taste) = 'B') or
[B64]  58Ø:         (XPos>39Ø) and (XPos<444) and
[884]  581:         (YPos>66) and (YPos<76) then
[MUØ]  582:      begin
[SM1]  583:        MouseAus;
[YC2]  584:        WA (2Ø6, 4Ø,1,'is');
[PL4]  585:        EndSprite := LiesT (246,254,4Ø,
[LM1]  586:                            'ASprite',
[PY1]  587:                            EndSprite);
[2D2]  588:        WA (2Ø6, 4Ø,3,'is');
[S36]  589:        SpriteNr := Ord (BeginSprite[1])-64;
[TF1]  59Ø:        MouseEin
[ZNØ]  591:      end;
[ØØØ]  592:
[T24]  593:      if (UpCase (Taste) = 'X') or
[X54]  594:         (XPos>39Ø) and (XPos<412) and
[CC4]  595:         (YPos>98) and (YPos<1Ø5) then
[MUØ]  596:      begin
[SM1]  597:        MouseAus;
[5A2]  598:        SiCo := StepX;
[NY2]  599:        Str (StepX,Help);
[2K3]  6ØØ:        Help := LiesT (23Ø,246,7Ø,
[N62]  6Ø1:                       'AStep',Help);
[3Y3]  6Ø2:        Val (Help,StepX,Check);
[GK3]  6Ø3:        if (Abs (StepX) > 3) or
[3D2]  6Ø4:           (Check <> Ø) then
[MUØ]  6Ø5:        begin
[A92]  6Ø6:          StepX := SiCo;
[NY2]  6Ø7:          Str (StepX,Help);
[TM3]  6Ø8:          FillBox (23Ø,7Ø,246,77,Ø);
[EC2]  6Ø9:          WA (23Ø,7Ø,3,Help)
[ZNØ]  61Ø:        end;
[TF1]  611:        MouseEin
[ZNØ]  612:      end;
[ØØØ]  613:
[X24]  614:      if (UpCase (Taste) = 'Y') or
[X54]  615:         (XPos>39Ø) and (XPos<412) and
[FF4]  616:         (YPos>1Ø8) and (YPos<117) then
[MUØ]  617:      begin
[SM1]  618:        MouseAus;
[7A2]  619:        SiCo := StepY;
[SY2]  62Ø:        Str (StepY,Help);
[3K3]  621:        Help := LiesT (23Ø,246,8Ø,
[N62]  622:                       'AStep',Help);
[7Y3]  623:        Val (Help,StepY,Check);
[LK3]  624:        if (Abs (StepY) > 3) or
[3D2]  625:           (Check <> Ø) then
[MUØ]  626:        begin
[C92]  627:          StepY := SiCo;
[SY2]  628:          Str (StepY,Help);
[YM3]  629:          FillBox (23Ø,8Ø,246,87,Ø);
[GC2]  63Ø:          WA (23Ø,8Ø,3,Help)
[ZNØ]  631:        end;
[TF1]  632:        MouseEin
[ZNØ]  633:      end;
[ØØØ]  634:
[NØ4]  635:      if (UpCase (Taste) = 'G') or
[C64]  636:         (XPos>374) and (XPos<615) and
[PF4]  637:         (YPos>127) and (YPos<136) then
[MUØ]  638:      begin
[SM1]  639:        MouseAus;
[UK4]  64Ø:        WA (198,1ØØ,1,'eschwindigkeit');
[VH1]  641:        SiCo := V;
[152]  642:        Str (V,Help);
[UR3]  643:        Help := LiesT (262,27Ø,11Ø,
[7T1]  644:                       'AV',Help);
[153]  645:        Val (Help,V,Check);
[YK4]  646:        WA (198,1ØØ,3,'eschwindigkeit');
[TF1]  647:        MouseEin
[ZNØ]  648:      end;
[ØØØ]  649:
```

```
[MA5]  65Ø:      if SpriteNr = Ord (EndSprite[1])
[381]  651:                        - 64 then
[HL5]  652:        SpriteNr := Ord (BeginSprite[1])
[6EØ]  653:                        - 64
[ZRØ]  654:      else
[E93]  655:      if SpriteNr = 12 then
[2E5]  656:        SpriteNr := 1 else Inc (SpriteNr);
[ØØØ]  657:
[9P4]  658:      SpriteX := SpriteX + StepX;
[LP4]  659:      SpriteY := SpriteY + StepY;
[SL4]  66Ø:      PutImage (SpriteX,SpriteY,
[BK3]  661:                Sprite[SpriteNr],Ø);
[482]  662:      Delay (45Ø-V*5Ø);
[SL4]  663:      PutImage (SpriteX,SpriteY,
[XC2]  664:                Sprite[14],Ø);
[ØØØ]  665:
[WL4]  666:      if (SpriteX + StepX < 1) then
[YE2]  667:        SpriteX := 14Ø;
[MW4]  668:      if (SpriteX + StepX > 14Ø) then
[G32]  669:        SpriteX := 1;
[ØØØ]  67Ø:
[SM4]  671:      if (SpriteY + StepY < 6) then
[YE2]  672:        SpriteY := 12Ø;
[UW4]  673:      if (SpriteY + StepY > 12Ø) then
[3Ø2]  674:        SpriteY := 6
[493]  675:    until (Taste = #27) or
[B64]  676:          (XPos>374) and (XPos<426) and
[BU3]  677:          (YPos>157) and (YPos<167);
[2K1]  678:    XPos := Ø;
[4K1]  679:    YPos := Ø;
[PS4]  68Ø:    EditierenScreen (True,True)
[UD4]  681: end (* SequenzAnimation *);
[ØØØ]  682:
[LL3]  683: {*** S P E I C H E R N ***}
[ØØØ]  684:
[L66]  685: procedure FehlerCheck (Art : String7Ø;
[X63]  686:                        var Status : Byte);
[P33]  687:  var SZ :    Integer;
[712]  688:      Taste : Char;
[PF3]  689: begin (* FehlerCheck *)
[WV5]  69Ø:   if (Status <> Ø) and (Status <> 2) then
[MUØ]  691:   begin
[FU3]  692:     FillBox (16,1ØØ,14Ø,156,Ø);
[2K3]  693:     WA (17,1Ø1,3,'ACHTUNG !!!');
[UM3]  694:     OutTextXY (17,116,Art);
[945]  695:     OutTextXY (17,131,'Wiederholen');
[P54]  696:     OutTextXY (17,146,'(j/n) ?');
[HR2]  697:     Taste := ReadKey;
[884]  698:     if UpCase (Taste) <> 'J' then
[TR1]  699:       Status := 2
[ZRØ]  7ØØ:     else
[MUØ]  7Ø1:     begin
[FU3]  7Ø2:       FillBox (16,1ØØ,14Ø,156,Ø);
[NR1]  7Ø3:       Status := 1
[4MØ]  7Ø4:     end
[4MØ]  7Ø5:   end
[N73]  7Ø6: end (* FehlerCheck *);
[ØØØ]  7Ø7:
[R16]  7Ø8: procedure Speichern (Was : String7Ø);
[K94]  7Ø9:  var Datei :      String[8];
[UC2]  71Ø:      Feil :       Text;
[2A3]  711:      SeqZ, Beit : Integer;
[GM2]  712:      Status :     Byte;
[WF2]  713:      Taste :      Char;
[643]  714: begin (* Speichern *)
[SM1]  715:   MouseAus;
[B85]  716:   GetImage (1Ø3,6,142,45,Sprite[13]);
[2A3]  717:   FillBox (Ø,1,15Ø,17Ø,Ø);
[ØØØ]  718:
[BU4]  719:   WA       (17,7,3,'Unter  welchem');
[RZ4]  72Ø:   OutTextXY (17,19,'Namen soll ');
[KC3]  721:   if Was = 'Sprite' then
[MUØ]  722:   begin
[FP3]  723:     OutTextXY (1Ø5,19,'der');
[2P4]  724:     OutTextXY (17,34,'    Sprite');
[K83]  725:     WA (16,33,2,'    Sprite')
[ZNØ]  726:   end;
[ØØØ]  727:
[GK3]  728:   if Was = 'Sequenz' then
[MUØ]  729:   begin
[GN3]  73Ø:     OutTextXY (1Ø5,19,'die');
[TR4]  731:     OutTextXY (17,34,'    Sequenz');
[YC3]  732:     WA (16,33,2,'    Sequenz')
[ZNØ]  733:   end;
[ØØØ]  734:
[4Ø4]  735:   WA (17,46,3,' gespeichert');
[2E4]  736:   OutTextXY (17,61,'werden :');
[VZ3]  737:   WA (16,45,2,' gespeichert');
[ØØØ]  738:
[KC3]  739:   if Was = 'Sprite' then
[AC4]  74Ø:     WA (17,75,3,'________.SPR')
[ZRØ]  741:   else
[SD4]  742:     WA (17,75,3,'________.SEQ');
```

Listing. »spred.pas« macht das Arbeiten mit Sprites einfach

```
[ØØØ]  743:
[V45]  744:   Datei := LiesT (17,81,74,'Datei','');
[XT2]  745:   if Datei <> '' then
[P31]  746:     repeat
[KC3]  747:       if Was = 'Sprite' then
[Z24]  748:         Assign (Feil,Datei+'.SPR')
[ZRØ]  749:       else
[Y84]  75Ø:         Assign (Feil,Datei+'.SEQ');
[864]  751:       {$I-} Reset (Feil); {$I+}
[S93]  752:       Status := IOResult;
[DT2]  753:       if Status = Ø then
[MUØ]  754:       begin
[EP4]  755:        WA (17,1Ø1,3,      'Soll die alte,');
[WG5]  756:        OutTextXY (17,113,' gleichnamige');
[PF4]  757:        OutTextXY (17,125,'     Datei');
[ZK5]  758:        OutTextXY (17,137,'ueberschrieben');
[A65]  759:        OutTextXY (17,149,'werden (j/n) ?');
[KM4]  76Ø:        WA (16,136,2,'ueberschrieben');
[HR2]  761:        Taste := ReadKey;
[R34]  762:        if UpCase (Taste) = 'J' then
[TR1]  763:          Status := 2
[ZNØ]  764:       end;
[ØØØ]  765:
[HT2]  766:       if Status = 2 then
[MUØ]  767:       begin
[DXØ]  768:         {$I-}
[1L2]  769:         ReWrite (Feil);
[KC3]  77Ø:         if Was = 'Sprite' then
[7H3]  771:           for Beit := 1 to 4Ø4 do
[E35]  772:             WriteLn (Feil,Sprite[13,Beit])
[ZRØ]  773:         else
[7C3]  774:           for SeqZ := 1 to 12 do
[7H3]  775:             for Beit := 1 to 4Ø4 do
[TL4]  776:               WriteLn (Feil,Sprite[SeqZ,
[C71]  777:                        Beit]);
[XZ1]  778:         Close (Feil);
[BXØ]  779:         {$I+}
[C23]  78Ø:         Status := IOResult
[ZNØ]  781:       end;
[556]  782:       FehlerCheck ('Ausgabefehler',Status)
[GR3]  783:     until Status in [Ø,2];
[ØØØ]  784:
[2A3]  785:   FillBox (Ø,1,15Ø,17Ø,Ø);
[P12]  786:   Taste := Nix;
[R15]  787:   EditierenScreen (False,False);
[TF1]  788:   MouseEin
[DX2]  789: end (* Speichern *);
[ØØØ]  79Ø:
[PM2]  791: {*** L A D E N ***}
[ØØØ]  792:
[A95]  793: procedure Laden (Was : String7Ø);
[K94]  794:   var Datei :      String[8];
[UC2]  795:       Feil :       Text;
[2A3]  796:       SeqZ, Beit : Integer;
[GM2]  797:       Status :     Byte;
[WF2]  798:       Taste :      Char;
[UA2]  799: begin (* Laden *)
[SM1]  8ØØ:   MouseAus;
[B85]  8Ø1:   GetImage (1Ø3,6,142,45,Sprite[13]);
[2A3]  8Ø2:   FillBox (Ø,1,15Ø,17Ø,Ø);
[W84]  8Ø3:   WA       (17,7,3,'Welche Datei');
[X25]  8Ø4:   OutTextXY (17,19,'soll geladen');
[PD4]  8Ø5:   OutTextXY (17,31,'werden :');
[P63]  8Ø6:   WA (56,18,2,'geladen');
[ØØØ]  8Ø7:
[KC3]  8Ø8:   if Was = 'Sprite' then
[9C4]  8Ø9:     WA (17,46,3,'________.SPR')
[ZRØ]  81Ø:   else
[RD4]  811:     WA (17,46,3,'________.SEQ');
[ØØØ]  812:
[G45]  813:   Datei := LiesT (17,81,45,'Datei','');
[XT2]  814:   if Datei <> '' then
[P31]  815:     repeat
[KC3]  816:       if Was = 'Sprite' then
[Z24]  817:         Assign (Feil,Datei+'.SPR')
[ZRØ]  818:       else
[Y84]  819:         Assign (Feil,Datei+'.SEQ');
[DXØ]  82Ø:       {$I-}
[612]  821:       Reset (Feil);
[S93]  822:       Status := IOResult;
[DT2]  823:       if Status = Ø then
[MUØ]  824:       begin
[KC3]  825:         if Was = 'Sprite' then
[7H3]  826:           for Beit := 1 to 4Ø4 do
[1R4]  827:             ReadLn (Feil,Sprite[13,Beit])
[ZRØ]  828:         else
[7C3]  829:           for SeqZ := 1 to 12 do
[7H3]  83Ø:             for Beit := 1 to 4Ø4 do
[V74]  831:               ReadLn (Feil,Sprite[SeqZ,
[C71]  832:                       Beit]);
[XZ1]  833:         Close (Feil);
[BXØ]  834:         {$I+}
[C23]  835:         Status := IOResult
[ZNØ]  836:       end;
[ØØØ]  837:
[HT2]  838:       if Status = 2 then
[MUØ]  839:       begin
[7E4]  84Ø:         WA (17,1Ø1,3,'Genannte  Datei');
[N85]  841:         OutTextXY (17,113,'   existiert');
[BG4]  842:         OutTextXY (17,125,'    nicht !');
[3Z3]  843:         WA (16,112,2,'   existiert');
[KE3]  844:         WA (16,124,2,'    nicht !');
[HR2]  845:         Taste := ReadKey;
[HR1]  846:         Status := Ø
[ZNØ]  847:       end;
[9G5]  848:       FehlerCheck ('Lesefehler',Status)
[GR3]  849:     until Status in [Ø,2];
[2A3]  85Ø:   FillBox (Ø,1,15Ø,17Ø,Ø);
[P12]  851:   Taste := Nix;
[1X4]  852:   EditierenScreen (False,True);
[TF1]  853:   MouseEin
[B42]  854: end (* Laden *);
[ØØØ]  855:
[ØM3]  856: {*** Q U E L L C O D E ***}
[ØØØ]  857:
[216]  858: procedure QuellCode (Was : String7Ø);
[XU4]  859:  var HelpSprite : SpriteTyp;
[VK3]  86Ø:      Datei :      String[8];
[U44]  861:      GroesseStr : String[3];
[AF3]  862:      Regs :       Registers;
[UC2]  863:      Feil :       Text;
[LW2]  864:      GroesseInt, SPZ,
[PU1]  865:      SeqZ, Beit,
[Y93]  866:      HelpFarbe  : Integer;
[2H1]  867:      LOX, LOY,
[LD1]  868:      SZ, RUX,
[8D3]  869:      RUY, Status : Byte;
[NM2]  87Ø:      Taste :      Char;
[M33]  871: begin (* QuellCode *)
[SM1]  872:   MouseAus;
[B85]  873:   GetImage (1Ø3,6,142,45,Sprite[13]);
[2A3]  874:   FillBox (Ø,1,15Ø,17Ø,Ø);
[H24]  875:   WA (17,7,3,'Unter welchem');
[RZ4]  876:   OutTextXY (17,19,'Namen soll ');
[KC3]  877:   if Was = 'Sprite' then
[TK3]  878:     OutTextXY (1Ø5,19,'der')
[ZRØ]  879:   else
[GN3]  88Ø:     OutTextXY (1Ø5,19,'die');
[KC3]  881:   if Was = 'Sprite' then
[MP5]  882:     OutTextXY (17,34,'Sprite als Unit')
[ZRØ]  883:   else
[6W5]  884:     OutTextXY (17,34,'Sequenz als Unit');
[T15]  885:   OutTextXY (17,48,'gespeichert');
[1E4]  886:   OutTextXY (17,6Ø,'werden :');
[KC3]  887:   if Was = 'Sprite' then
[E44]  888:     WA (16,33,2,'Sprite     Unit')
[ZRØ]  889:   else
[HC4]  89Ø:     WA (16,33,2,'Sequenz     Unit');
[GD4]  891:   WA (17,77,3,'________.PAS');
[X45]  892:   Datei := LiesT (17,81,76,'Datei','');
[ØØØ]  893:
[3Y2]  894:   if (Datei <> '') then
[MUØ]  895:   begin
[2A3]  896:     FillBox (Ø,1,15Ø,17Ø,Ø);
[434]  897:     WA       (17,7,3,'Klicken Sie');
[HE5]  898:     OutTextXY (17,19,'nun die linke,');
[HD5]  899:     OutTextXY (17,31,'obere und die');
[LN5]  9ØØ:     OutTextXY (17,43,'rechte, untere');
[4B4]  9Ø1:     OutTextXY (17,55,'Ecke des');
[HA5]  9Ø2:     OutTextXY (17,67,'Ausschnittes');
[MH3]  9Ø3:     OutTextXY (17,79,'an.');
[FL1]  9Ø4:     MouseEin;
[PA1]  9Ø5:     LOX := Ø;
[UA1]  9Ø6:     LOY := Ø;
[MB1]  9Ø7:     RUX := Ø;
[SB1]  9Ø8:     RUY := Ø;
[CT1]  9Ø9:     Status := Ø;
[3C2]  91Ø:     HelpFarbe := Ø;
[P31]  911:     repeat
[2K1]  912:       XPos := Ø;
[4K1]  913:       YPos := Ø;
[ØØØ]  914:
[NY1]  915:       Regs.AX := 3;
[PB2]  916:       Intr ($33,Regs);
[YK3]  917:       if Regs.BX and 3 > Ø then
[MUØ]  918:       begin
[LP2]  919:         XPos := Regs.CX;
[SP2]  92Ø:         YPos := Regs.DX;
[C44]  921:         if Xpos < 3ØØ then Delay(2ØØ)
[ZNØ]  922:       end;
[ØØØ]  923:
[P12]  924:       Taste := Nix;
[Y23]  925:       if KeyPressed then
[MUØ]  926:       begin
[HR2]  927:         Taste := ReadKey;
[SG5]  928:         if Taste = #Ø then Taste := ReadKey
```

```
[ZNØ]  929:        end;
[ØØØ]  93Ø:
[CW4]  931:        if (XPos>=314) and (XPos<=63Ø) and
[SH4]  932:           (YPos>=35) and (YPos<193) then
[7H2]  933:          case Status of
[W81]  934:          Ø : begin
[SM1]  935:                MouseAus;
[LK4]  936:                HelpFarbe := GetPixel ((XPos
[WK3]  937:                 div 2 -156) div 4 * 4 + 157,
[EH3]  938:                 (YPos -35)  div 4 * 4 + 6);
[GT1]  939:                Status := 1;
[ZH3]  94Ø:                LOX := (XPos div 2 - 156)
[2C1]  941:                 div 4 + 1;
[ØV3]  942:                LOY := (YPos - 35) div 4 + 1;
[B63]  943:                if HelpFarbe = 2 then
[5Y1]  944:                 SetColor (3)
[BV2]  945:                else SetColor (2);
[XW2]  946:                for SZ := 1 to 3 do
[DE3]  947:                 Line ((XPos div 2 -156)
[4E3]  948:                  div 4 * 4 + 157,(YPos -35)
[872]  949:                  div 4 * 4 + 5+SZ,
[3C2]  95Ø:                  (XPos div 2 -156)
[CE3]  951:                  div 4 * 4 + 159,(YPos -35)
[LC2]  952:                  div 4 * 4 + 5+SZ);
[FL1]  953:                MouseEin;
[ØK1]  954:                Delay (2ØØ)
[ZNØ]  955:               end;
[Y81]  956:          1 : begin
[XK3]  957:                RUX := (XPos div 2 - 156)
[2C1]  958:                 div 4 + 1;
[AM2]  959:                RUY := (YPos - 35)
[2C1]  96Ø:                 div 4 + 1;
[WP2]  961:                if (RUX = LOX) and
[5G2]  962:                   (RUY = LOY) then
[MUØ]  963:                begin
[SM1]  964:                 MouseAus;
[BE3]  965:                 FillBox ((LOX-1)*4+157,
[MR2]  966:                  (LOY-1)*4+6,(LOX-1)*
[H82]  967:                  4+159,(LOY-1)*4+8,
[9X1]  968:                  HelpFarbe);
[FL1]  969:                 MouseEin;
[PA1]  97Ø:                 LOX := Ø;
[UA1]  971:                 LOY := Ø;
[MB1]  972:                 RUX := Ø;
[SB1]  973:                 RUY := Ø;
[CT1]  974:                 Status := Ø;
[ØK1]  975:                 Delay (2ØØ)
[4MØ]  976:                end
[TN2]  977:                else Status := 2
[ZNØ]  978:               end;
[4MØ]  979:          end
[U35]  98Ø:     until (Status = 2) or (Taste = #27);
[ØØØ]  981:
[SM1]  982:     MouseAus;
[A35]  983:     FillBox ((LOX-1)*4+157,(LOY-1)*4+6,
[MC3]  984:              (LOX-1)*4+159,(LOY-1)*4+8,
[9X1]  985:              HelpFarbe);
[ØØØ]  986:
[EØ3]  987:     if Taste <> #27 then
[P31]  988:       repeat
[784]  989:         Assign (Feil,Datei+'.PAS');
[864]  99Ø:         {$I-} Reset (Feil); {$I+}
[S93]  991:         Status := IOResult;
[TW2]  992:         if  Status = Ø then
[MUØ]  993:         begin
[2H3]  994:           FillBox (Ø,1ØØ,15Ø,17Ø,Ø);
[FH1]  995:           WA (17,1Ø1,3,
[BN2]  996:               'Soll die alte,');
[FT2]  997:           OutTextXY (17,113,
[3K2]  998:                      ' gleichnamige');
[MT2]  999:           OutTextXY (17,125,
[4S1] 1ØØØ:                      '     Datei');
[TT2] 1ØØ1:           OutTextXY (17,137,
[P23] 1ØØ2:                      'ueberschrieben');
[YT2] 1ØØ3:           OutTextXY (17,149,
[SL2] 1ØØ4:                      'werden (j/n) ?');
[KM4] 1ØØ5:           WA (16,136,2,'ueberschrieben');
[HR2] 1ØØ6:           Taste := ReadKey;
[R34] 1ØØ7:           if UpCase (Taste) = 'J' then
[TR1] 1ØØ8:             Status := 2
[ZNØ] 1ØØ9:         end;
[ØØØ] 1Ø1Ø:
[HT2] 1Ø11:         if Status = 2 then
[MUØ] 1Ø12:         begin
[DXØ] 1Ø13:           {$I-}
[1L2] 1Ø14:           ReWrite (Feil);
[N35] 1Ø15:           WriteLn (Feil,'Unit '+Datei+';');
[ND4] 1Ø16:           WriteLn (Feil,'Interface');
[MP4] 1Ø17:           if Frac((RUX-LOX+1) / 4) > Ø then
[ØU4] 1Ø18:             GroesseInt := ((Abs (RUX-LOX)+
[NN1] 1Ø19:              1) div 4 + 1) *
[BV2] 1Ø2Ø:              (Abs (RUY-LOY)+1) + 4
[ZRØ] 1Ø21:           else
```

```
[ØU4] 1Ø22:             GroesseInt := ((Abs (RUX-LOX)+
[3A1] 1Ø23:              1) div 4) *
[WØ3] 1Ø24:              (Abs (RUY-LOY)+1) + 4;
[U25] 1Ø25:           Str (GroesseInt,GroesseStr);
[KC3] 1Ø26:           if Was = 'Sprite' then
[MUØ] 1Ø27:           begin
[9H3] 1Ø28:             FillBox (Ø,121,15Ø,17Ø,Ø);
[SX2] 1Ø29:             Box (49,124,91,165,3);
[WY4] 1Ø3Ø:             PutImage (5Ø,125,Sprite[13],Ø);
[W74] 1Ø31:             GetImage (49+LOX,124+LOY,49+
[N22] 1Ø32:                       RUX,124+RUY,
[W22] 1Ø33:                       HelpSprite);
[655] 1Ø34:             WriteLn (Feil,'Const Sprite :'+
[DZ1] 1Ø35:                      ' Array [1..'+
[N22] 1Ø36:                     GroesseStr+
[852] 1Ø37:                     '] of Byte = (');
[FB1] 1Ø38:             SPZ := Ø;
[G15] 1Ø39:             for Beit := 1 to GroesseInt do
[MUØ] 1Ø4Ø:             begin
[EM1] 1Ø41:               Inc (SPZ);
[6P4] 1Ø42:               Write (Feil,Ord (HelpSprite
[VE1] 1Ø43:                                [Beit]));
[X34] 1Ø44:               if Beit = GroesseInt then
[HY2] 1Ø45:                 WriteLn (Feil,');')
[6B3] 1Ø46:               else Write (Feil,',');
[X92] 1Ø47:               if SPZ = 3Ø then
[MUØ] 1Ø48:               begin
[ØL2] 1Ø49:                 WriteLn (Feil);
[T71] 1Ø5Ø:                 SPZ := Ø
[4MØ] 1Ø51:               end
[4MØ] 1Ø52:             end
[4MØ] 1Ø53:           end
[ZRØ] 1Ø54:           else
[MUØ] 1Ø55:           begin
[655] 1Ø56:             WriteLn (Feil,'Const Sprite :'+
[4N2] 1Ø57:                      ' Array [1..12,1..'+
[N22] 1Ø58:                     GroesseStr+
[852] 1Ø59:                     '] of Byte = (');
[7C3] 1Ø6Ø:             for SeqZ := 1 to 12 do
[MUØ] 1Ø61:             begin
[9H3] 1Ø62:               FillBox (Ø,121,15Ø,17Ø,Ø);
[SX2] 1Ø63:               Box (49,124,91,165,3);
[KF2] 1Ø64:               PutImage (5Ø,125,
[6V2] 1Ø65:                         Sprite[SeqZ],Ø);
[W74] 1Ø66:               GetImage (49+LOX,124+LOY,49+
[N22] 1Ø67:                         RUX,124+RUY,
[W22] 1Ø68:                         HelpSprite);
[FB1] 1Ø69:               SPZ := Ø;
[ED2] 1Ø7Ø:               Write (Feil,'(');
[XK4] 1Ø71:               for Beit:=1 to GroesseInt do
[MUØ] 1Ø72:               begin
[EM1] 1Ø73:                Inc (SPZ);
[6P4] 1Ø74:                Write (Feil,Ord (HelpSprite
[VE1] 1Ø75:                                  [Beit]));
[284] 1Ø76:                if (Beit = GroesseInt) then
[MUØ] 1Ø77:                begin
[9K2] 1Ø78:                 if SeqZ = 12 then
[YG2] 1Ø79:                   Write (Feil,'));')
[HG3] 1Ø8Ø:                 else Write (Feil,'),');
[4MØ] 1Ø81:                end
[6B3] 1Ø82:                else Write (Feil,',');
[X92] 1Ø83:                 if SPZ = 3Ø then
[MUØ] 1Ø84:                 begin
[ØL2] 1Ø85:                  WriteLn (Feil);
[T71] 1Ø86:                  SPZ := Ø
[4MØ] 1Ø87:                 end;
[ZNØ] 1Ø88:               end;
[MB2] 1Ø89:               WriteLn (Feil)
[4MØ] 1Ø9Ø:             end;
[ZNØ] 1Ø91:           end;
[R95] 1Ø92:           WriteLn (Feil,'Implementation');
[8D3] 1Ø93:           WriteLn (Feil,'end.');
[XZ1] 1Ø94:           Close (Feil);
[BXØ] 1Ø95:           {$I+}
[C23] 1Ø96:           Status := IOResult
[ZNØ] 1Ø97:         end;
[ZR4] 1Ø98:         FehlerCheck ('Ausgabefehler',
[1A1] 1Ø99:                      Status)
[ZH3] 11ØØ:       until Status in [Ø,2]
[ZNØ] 11Ø1:   end;
[2A3] 11Ø2:   FillBox (Ø,1,15Ø,17Ø,Ø);
[P12] 11Ø3:   Taste := Nix;
[2K1] 11Ø4:   XPos := Ø;
[4K1] 11Ø5:   YPos := Ø;
[FL1] 11Ø6:   MouseEin;
[6U2] 11Ø7:   SetColor (Farbe);
[6W4] 11Ø8:   EditierenScreen (False,False)
[XW2] 11Ø9: end (* QuellCode *);
[ØØØ] 111Ø:
[6Z3] 1111: {*** E N D E A B F R A G E ***}
[ØØØ] 1112:
[8M4] 1113: function EndeAbfrage : Char;
[Y92] 1114:  var SZ : Byte;
```

Listing. »spred.pas« macht das Arbeiten mit Sprites einfach

```
[EE3] 1115: begin (* EndeAbfrage *)
[SM1] 1116:   MouseAus;
[UK3] 1117:   FillBox (2,149,140,157,0);
[9M4] 1118:   WA (3,150,3,'Sind Sie sicher ?');
[CT3] 1119:   EndeAbfrage := ReadKey;
[UK3] 1120:   FillBox (2,149,140,157,0);
[463] 1121:   WA (3,150,3,'Beenden');
[K12] 1122:   WA (2,149,2,'B');
[TF1] 1123:   MouseEin
[X63] 1124: end (* EndeAbfrage *);
[000] 1125:
[KW3] 1126: {*** H A U P T M E N Ü ***}
[000] 1127:
[VR2] 1128: procedure Menu;
[084] 1129:  var  H,P,F :         String[2];
[0C3] 1130:       Regs :         Registers;
[D53] 1131:       SZ, SZ2, SZ3 : Byte;
[PL2] 1132:       Taste :        Char;
[272] 1133: begin (* Menu *)
[P31] 1134:   repeat
[2K1] 1135:     XPos := 0;
[4K1] 1136:     YPos := 0;
[NY1] 1137:     Regs.AX := 3;
[PB2] 1138:     Intr ($33,Regs);
[YK3] 1139:     if Reqs.BX and 3 > 0 then
[MU0] 1140:     begin
[LP2] 1141:       XPos := Regs.CX;
[SP2] 1142:       YPos := Regs.DX;
[C44] 1143:       if Xpos < 300 then Delay(200)
[ZN0] 1144:     end;
[000] 1145:
[P12] 1146:     Taste := Nix;
[Y23] 1147:     if KeyPressed then
[MU0] 1148:     begin
[HR2] 1149:       Taste := ReadKey;
[SG5] 1150:       if Taste = #0 then Taste := ReadKey
[ZN0] 1151:     end;
[000] 1152:
[914] 1153:     if (UpCase (Taste) = 'L') or
[C54] 1154:        (XPos>38) and (XPos<122) and
[D74] 1155:        (YPos>43) and (YPos<52) then
[CK2] 1156:       Laden ('Sprite');
[524] 1157:     if (UpCase (Taste) = 'S') or
[E64] 1158:        (XPos>38) and (XPos<186) and
[L74] 1159:        (YPos>53) and (YPos<62) then
[LA3] 1160:       Speichern ('Sprite');
[X14] 1161:     if (UpCase (Taste) = 'Q') or
[E64] 1162:        (XPos>38) and (XPos<186) and
[T74] 1163:        (YPos>63) and (YPos<72) then
[CB3] 1164:       QuellCode ('Sprite');
[904] 1165:     if (UpCase (Taste) = 'D') or
[C54] 1166:        (XPos>38) and (XPos<122) and
[G84] 1167:        (YPos>85) and (YPos<95) then
[7W2] 1168:       Laden ('Sequenz');
[D04] 1169:     if (UpCase (Taste) = 'E') or
[E64] 1170:        (XPos>38) and (XPos<186) and
[AC4] 1171:        (YPos>96) and (YPos<105) then
[EM3] 1172:       Speichern ('Sequenz');
[D24] 1173:     if (UpCase (Taste) = 'U') or
[E64] 1174:        (XPos>38) and (XPos<186) and
[9F4] 1175:        (YPos>106) and (YPos<115) then
[7N3] 1176:       QuellCode ('Sequenz');
[154] 1177:     if (UpCase(Taste) = 'M') or
[T54] 1178:        (XPos>38) and (XPos<234) and
[EF4] 1179:        (YPos>116) and (YPos<125) then
[ZK2] 1180:       SequenzScreen;
[X34] 1181:     if (UpCase(Taste) = 'A') or
[T54] 1182:        (XPos>38) and (XPos<234) and
[LF4] 1183:        (YPos>126) and (YPos<135) then
[U43] 1184:       SequenzAnimation;
[000] 1185:
[A54] 1186:     if (Upcase(Taste) = 'P') or
[L84] 1187:        (XPos>0) and (XPos<116) and
[VF4] 1188:        (YPos>137) and (Ypos<146) then
[MU0] 1189:     begin
[SM1] 1190:       MouseAus;
[6T3] 1191:       FillBox (115,110,123,127,0);
[9F3] 1192:       if PalettenNr = 3 then
[4E2] 1193:         PalettenNr := 0
[UP3] 1194:       else Inc (PalettenNr);
[8P3] 1195:       Palette (PalettenNr);
[PD3] 1196:       Str (PalettenNr,P);
[S52] 1197:       WA (115,110,3,P);
[ZD3] 1198:       OutTextXY (115,120,'0');
[UR2] 1199:       Hintergrund := 0;
[6U2] 1200:       SetColor (Farbe);
[TF1] 1201:       MouseEin
[ZN0] 1202:     end;
[000] 1203:
[K44] 1204:     if (Upcase(Taste) = 'H') or
[894] 1205:        (XPos>0) and (XPos<179) and
[0G4] 1206:        (YPos>147) and (Ypos<156) then
[MU0] 1207:     begin
[SM1] 1208:       MouseAus;
[CT3] 1209:       FillBox (115,120,131,127,0);
[1T3] 1210:       if Hintergrund = 15 then
[4M2] 1211:         Hintergrund := 0
[9Y3] 1212:       else Inc (Hintergrund);
[KA4] 1213:       SetBkColor (Hintergrund);
[SE3] 1214:       Str (Hintergrund,H);
[V42] 1215:       WA (115,120,3,H);
[6U2] 1216:       SetColor (Farbe);
[TF1] 1217:       MouseEin
[ZN0] 1218:     end;
[000] 1219:
[C44] 1220:     if (Upcase(Taste) = 'F') or
[2Z3] 1221:        (XPos>0) and (XPos<84) and
[5G4] 1222:        (YPos>157) and (Ypos<166) then
[MU0] 1223:     begin
[SM1] 1224:       MouseAus;
[MT3] 1225:       FillBox (115,130,131,137,0);
[PE2] 1226:       if Farbe = 3 then
[SD1] 1227:         Farbe := 0
[3P2] 1228:       else Inc (Farbe);
[E62] 1229:       Str (Farbe,F);
[LW2] 1230:       WA (115,130,Farbe,F);
[TF1] 1231:       MouseEin
[ZN0] 1232:     end;
[000] 1233:
[754] 1234:     if (Upcase(Taste) = 'O') or
[G84] 1235:        (XPos>2) and (XPos<131) and
[AG4] 1236:        (YPos>167) and (Ypos<176) then
[ZZ4] 1237:       if UpCase (EndeAbfrage) = 'J' then
[MU0] 1238:       begin
[SM1] 1239:         MouseAus;
[MD3] 1240:         FillBox (103,6,142,45,0);
[F12] 1241:         SetColor (0);
[NV2] 1242:         for SZ := 1 to 40 do
[B23] 1243:          for SZ2 := 1 to 40 do
[UG4] 1244:           for SZ3 := 1 to 3 do Line (153+
[W74] 1245:            SZ2*4,1+SZ*4+SZ3,153+SZ2*4+2,
[4N1] 1246:            1+SZ*4+SZ3);
[6U2] 1247:         SetColor (Farbe);
[TF1] 1248:         MouseEin
[ZN0] 1249:       end;
[000] 1250:
[CW4] 1251:     if (XPos>=314) and (XPos<=630) and
[SH4] 1252:        (YPos>=35) and (YPos<193) then
[MU0] 1253:     begin
[SM1] 1254:      MouseAus;
[XW2] 1255:      for SZ := 1 to 3 do
[CY4] 1256:       Line ((XPos div 2 -156) div 4 * 4 +
[DA4] 1257:          157,(YPos -35)  div 4 * 4 + 5+SZ,
[KT3] 1258:          (XPos div 2 -156) div 4 * 4 +
[0G4] 1259:          159,(YPos -35)  div 4 * 4 + 5+SZ);
[RA4] 1260:      PutPixel ((XPos div 2 - 156)
[1Z2] 1261:                div 4 +103,(YPos - 35)
[NE2] 1262:                div 4 + 6,Farbe);
[TF1] 1263:      MouseEin
[ZN0] 1264:     end;
[000] 1265:
[C15] 1266:     if (Upcase(Taste) in ['B',#27]) or
[L84] 1267:        (XPos>0) and (XPos<116) and
[FG4] 1268:        (YPos>177) and (YPos<186) then
[MU0] 1269:     begin
[ZZ4] 1270:       if UpCase (EndeAbfrage) = 'J' then
[2P1] 1271:         Taste := #27
[ZV2] 1272:       else Taste := Nix;
[4M0] 1273:     end
[A64] 1274:   until (Upcase(Taste)) = #27
[272] 1275: end (* Menu *);
[000] 1276:
[DM4] 1277: {*** H A U P T P R O G R A M M ***}
[000] 1278:
[CK3] 1279: begin (* MainProgram *)
[AV0] 1280:   Init;
[B12] 1281:   MainScreen;
[MZ4] 1282:   EditierenScreen (True,False);
[XU1] 1283:   InitMouse;
[FL1] 1284:   MouseEin;
[CV0] 1285:   Menu;
[SM1] 1286:   MouseAus;
[AE2] 1287:   TextMode (C80);
[N53] 1288:   TextBackGround (0);
[3E2] 1289:   TextColor (15);
[141] 1290:   ClrScr
[K83] 1291: end (* MainProgram *).
```

Listing. »spred.pas« macht das Arbeiten mit Sprites einfach

50 gute Gründe, jetzt auf TextMaker 2.0 umzusteigen.

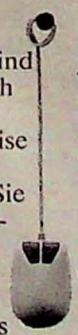
21. TextMaker 2.0 harmoniert mit allen gängigen Grafikkarten: CGA-Farbgrafik, Hercules, EGA, VGA meistert er spielend. Und bei EGA und VGA zeigt er seine Talente: Auf Wunsch läßt er Sie im 43- oder 50-Zeilenmodus arbeiten, damit Ihnen der Überblick über Ihre Texte nicht verlorengeht.

22. TextMaker ist ein sehr individuelles Programm. Nicht nur, daß er sich wichtige Systemeinstellungen bis zum nächsten Programmstart merkt: auch sämtliche Bildschirmfarben sind frei einstellbar. Wenn Ihnen also tatsächlich lila Menüs und rote Fenster gefallen: bitte sehr!

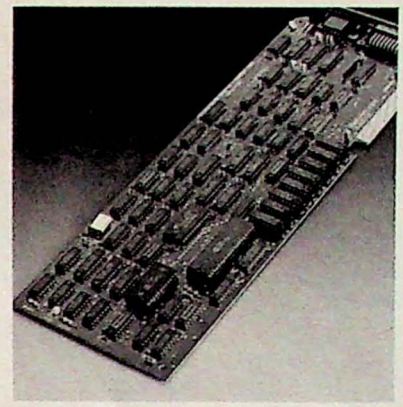
23. Blocksatz sorgt für den optisch perfekten Eindruck. TextMaker hilft nach: Sowohl feste als auch *proportionale* Schriften bringt er in Form. Ob Sie Flattersatz oder Blocksatz wählen – die Zeilen werden optimal aufgeteilt.

24. Zu TextMaker gehört ein klar verständliches deutsches Handbuch mit 263 Seiten Umfang. Sie müssen es aber nicht komplett lesen, weil es themenorientiert aufgebaut ist: Man liest nur Kapitel über die Funktionen, die man wirklich verwenden will.

25. Mode. Sie sind lich praktisch von Texten. sich wahlweise oder mittels So können Sie Cursorbewe- führen, zwi- stern hin- und Befehle down-Menüs



Mäuse sind in ja auch unheim- beim Editieren TextMaker läßt per Tastatur Maus bedienen. mit der Maus gungen durch- schen Textfen- und herspringen aus den Pull- auswählen.

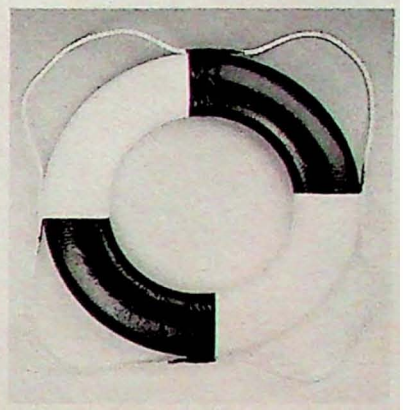


26. TextMaker nutzt den gesamten verfügbaren RAM für Ihre Texte. So packen Sie auch die größten Textbrocken. Wer Bücher mit TextMaker schreiben will, kann Dateien verketten und stößt damit allenfalls noch an die Kapazitätsgrenzen der Diskette oder Festplatte.

27. Die Silbentrennung – ein rotes Tuch für viele Textprogramme. Nicht so für TextMaker: Er besitzt eine ausgeklügelte deutsche Silbentrennung, deren Trennfehler im Promillebereich liegen. Wahlweise können Sie auch eine altgriechische (»Ma-gnet«) und eine Schweizer Silbentrennung zuschalten.

28. TextMaker besitzt eine sehr flexible Druckeransteuerung. Zahlreiche Druckertreiber für Matrix-, Typenrad- und Laserdrucker sind bereits im Lieferumfang enthalten, ständig werden neue erstellt. Und TextMaker kann sogar die bei vielen Druckern vorhandenen Sonderschriften wie »Orator«, »Sans-Serif«, »Outline« und »Shadow« ansteuern.

29. Versehentlich ein paar Zeilen oder einen ganzen Textabschnitt gelöscht? Kein Grund zur Panik! Auf Knopfdruck holt TextMaker die Zeilen wieder auf den Bildschirm zurück.



30. Der Preis? nur DM **248,–***

31.-50.

Lesen Sie nächsten Monat weiter – oder fordern Sie unseren Prospekt an, wenn Sie nicht so lange warten wollen.

* unverbindliche Preisempfehlung

Coupon

- ☐ Bitte schicken Sie mir erst einmal kostenlose Informationen.
- ☐ Senden Sie mir eine Demodiskette. Einen 10-Mark-Schein lege ich bei.
- ☐ Jawohl, senden Sie mir einen TextMaker ☐ Nachnahme ☐ Scheck liegt bei

Adresse:

Händleranfragen stets willkommen!

SoftMaker

SoftMaker GmbH · Grabbestraße 9 · 8500 Nürnberg 90 · Telefon 09 11 / 30 33 33

Das Tor zur Zukunft
Die Steckplätze des PC und AT

Das Steckkartenkonzept des PC und AT hat maßgeblich zum Erfolg dieser Computer beigetragen, denn durch die Steckplätze ist es möglich, eine bestehende Systemkonfiguration auszubauen und zu modernisieren. Dazu muß allerdings jeder Steckplatz alle wichtigen Signale der Mutterplatine zur Verfügung stellen.

Während viele andere Computersysteme schon nach zwei bis drei Jahren an Reiz verloren haben, gewinnt der PC ständig neue Anhänger. Die Tatsache, daß der PC immer noch – und gerade in letzter Zeit – attraktiv ist, verdankt er neben dem großen Softwareangebot dem Steckkartenkonzept. Durch die flexible Erweiterbarkeit auf Basis der Steckplätze, die man auch »Slots« nennt, halten ältere Geräte mit dem Stand der Technik Schritt. Neuentwicklungen sind durch die Steckplätze auch für die Besitzer von eher betagten Personalcomputern zugänglich. Durch einfaches Einstecken einer Erweiterungskarte verbessern Sie die Grafikauflösung oder erhöhen die Verarbeitungsgeschwindigkeit, ohne daß Sie gleich auf ein neues Computersystem umsteigen müssen und womöglich teure Software unbrauchbar wird.
Außerdem läßt sich ein PC durch dieses Steckkartenkonzept völlig problemlos an die unterschiedlichen Bedürfnisse des Anwenders anpassen. Während der eine Anwender auf eine qualitativ hochwertige Textdarstellung Wert legt, ist dem anderen hochauflösende Farbgrafik wichtiger. Durch das PC-Konzept eignet sich für beide Anwender der gleiche Computertyp, und sie müssen lediglich unterschiedliche Bildschirmkarten verwenden; das große Softwarepotential ist dagegen identisch.
Das Prinzip eines offenen Systems hat auch für den Einsteiger Vorteile: Er kann klein anfangen und sein Computersystem je nach Interesse und Finanzen beliebig ausbauen. Dabei spielt es keine Rolle, ob er eine Festplatte, zusätzliche Schnittstellen, eine Maus oder einen Joystick benötigt.

Die Steckplätze des PC sichern den Anschluß an die Zukunft

Die Steckplätze sind für den PC sozusagen »die Brücke zur Zukunft«, die dafür sorgt, daß der Computer mit den Ansprüchen des Anwenders mitwächst. Selbst der Käufer eines sehr alten PC mit 64 KByte Arbeitsspeicher und einer nicht grafikfähigen Bildschirmkarte kann sein Gerät durch einfachen Einbau von EGA-Karte, Speichererweiterung und eventuell einer 80286-(Beschleuniger-)-Karte dem heutigen Stand der

Technik anpassen. Derartige Erweiterungskarten sind nur deswegen denkbar, weil an einem Steckplatz nahezu alle Signale der Mutterplatine zur Verfügung stehen. Auf diese Weise kann die Peripherie auf einer Erweiterungskarte Informationen an den Prozessor und alle anderen Bausteine auf der Mutterplatine übergeben und von diesen empfangen. Für den Prozessor besteht zwischen den Bausteinen auf der Mutterplatine und den Bausteinen auf einer Steckkarte praktisch kein Unterschied.

Das Steckkartenkonzept von PC und AT hat entscheidend zum Erfolg dieser Computerfamilie beigetragen

Das Steckkartenkonzept des PC hat aber noch einen weiteren, nicht zu unterschätzenden Vorteil: Weil sich ein PC beliebig erweitern läßt und er trotzdem kompatibel bleibt, bilden alle PC-Modelle – von der Softwareseite her betrachtet – eine große Familie, die auf einen gemeinsamen Softwarepool zugreifen kann. Das macht MS-DOS und damit PCs allgemein für Softwareentwickler besonders attraktiv, denn wenn es von einem Computersystem weltweit mehr als zehn Millionen Exemplare gibt, sind die Verkaufschancen für ein gutes Programm natürlich viel größer als für jedes andere Computermodell.

Wie Sie sehen, ist der Steckplatz ein Faktor, der entscheidend zum heutigen Erfolg der IBM-kompatiblen Personalcomputer geführt hat. In letzter Zeit versuchen einige PC-Hersteller allerdings immer mehr Funktionen von den Steckkarten auf die Mutterplatine zu verlagern, und inzwischen sind sogar einige PCs auf dem Markt, die keinen einzigen Steckplatz enthalten.

Ob sich diese Art von Personalcomputern allerdings auf lange Sicht durchsetzen wird, ist mehr als zweifelhaft, denn der Verzicht auf die spätere Erweiterbarkeit ist sehr kurzsichtig, weil sich der Anwender dadurch die Zukunft verbaut.

Die Mehrzahl der PC-Hersteller hat dagegen erkannt, wie wichtig Steckplätze für einen Personalcomputer sind, und daher besitzen die meisten PCs auch zwischen drei und acht Steckplätze. In der Anfangszeit der PCs gab es von IBM sogar eine »IBM Expansion Unit«, die nur weitere Steckplätze und ein zusätzliches Netzteil enthielt. Allerdings ist diese Einheit nie in nennenswerten Stückzahlen verkauft worden.

Für den elektronischen Kontakt zwischen Steckplatz und Karte ist auf

Signal	E = Eingang A = Ausgang	Pin	Signalbeschreibung
*I/O CH CK	E	A1	Ein-/Ausgang-Steckplatzüberprüfung. Liegt diese Leitung auf 0, so signalisiert dieser Pegel für den Prozessor einen Adreßfehler oder Paritätsfehler auf einer der Steckkarten.
D0 bis D7	E/A	A2 bis A9	Datenleitungen 0 bis 7. Sie liefern die Daten für Prozessor, Speicher und Steckkarten. D0 bezeichnet das niedrigstwertige und D7 das höchstwertige Datenbit.
I/O CK RDY	E	A10	Steckkarte ist bereit. Im Normalfall liegt diese Leitung immer auf 1, das heißt, die Steckkarte ist zur Arbeit mit der normalen Systemgeschwindigkeit bereit. Langsamere Karten legen diesen Anschluß auf 0. Dadurch verlängert der Computer die Zyklen für Datenein- und -ausgabe, um auch der langsamen Karte den Zugriff auf das Computersystem zu erlauben. Die Maschinenzyklen werden in dieser Betriebsart um ein Vielfaches von 210 ns (ein Taktzyklus) verlängert. Die Leitung darf maximal zehn Taktzyklen lang auf 0 liegen.
AEN	A	A11	Mit Hilfe dieser Leitung kann der Computer über eine Karte auch dann noch DMA-Übertragungen durchführen, nachdem die Karte vom Prozessor und anderen Ein-/Ausgabe-Einheiten getrennt wurde. Bei aktiver Leitung kontrolliert die DMA-Steuerung den Adreß- und Datenbus sowie die Schreib- und Leseleitungen des Speichers.
A0 bis A19	A	A12 bis A31	Adreßleitungen 0 bis 19. Dies sind die zur Adressierung des Speichers und der Ein-/Ausgabe-Einheiten zuständigen Leitungen. Sie erlauben die Adressierung von 1 MByte Speicher und werden entweder vom Prozessor oder dem DMA-Controller erzeugt. A0 bezeichnet das niedrigstwertige und A19 das höchstwertige Adreßbit.
GND	–	B1	Masse. Dieser Anschluß dient als Bezugspotential für alle Signale und Versorgungsspannungen.
RESET DRV	A	B2	Über diesen Anschluß erfolgt die Neu-Initialisierung des Computersystems.
+5V	–	B3	Positive Versorgungsspannung vom Netzteil.
IRQ2/IRQ9**	E	B4	Interrupt-Anforderung an den Prozessor durch eine Karte. IRQ0 hat die höchste und IRQ7 die niedrigste Priorität. Nur die Systemuhr und die Tastatur des PC haben über die Leitungen IRQ0 und IRQ1 eine höhere Interrupt-Priorität als die IRQ-Signale der Steckkarten. Bei einer Interrupt-Anforderung geht der Anschluß auf 1 und verbleibt bis zur Quittierung durch den Prozessor (Acknowledge) in diesem Zustand.
-5V	–	B5	Negative Versorgungsspannung vom Netzteil.
DRQ2	E	B6	DMA-Anforderung. Asynchrone Anforderung eines Peripheriegeräts für den Zugriff auf eine DMA-Funktion. DRQ1 hat die höchste und DRQ3 die niedrigste Priorität. Bei einer Anforderung wechselt dieser Anschluß auf 1 und verbleibt bis zur Aktivierung der zugehörigen DACK-Leitung in diesem Zustand.
-12V	–	B7	Negative Spannung vom Netzteil.
*0WS/Card Select	E	B8	Zero-Waitstate. Bei einem AT zeigt eine Steckkarte dem Prozessor über diese Leitung an, daß er den derzeitigen Buszyklus ohne das Einfügen von zusätzlichen Wartezyklen ausführen darf. Bei einem Original-PC liegt an dieser Leitung nur am achten Steckplatz das Kartenauswahlsignal an.
+12V	–	B9	Positive Spannung vom Netzteil.
GND	–	B10	Masse. Dieser Anschluß dient als Bezugspotential für alle Signale und Versorgungsspannungen.
*MEMW	A	B11	Speicher-Schreibbefehl. Dieses Signal veranlaßt den Arbeitsspeicher des PC, die auf dem Datenbus anliegenden Daten zu speichern. Das Signal wird vom Prozessor oder der DMA-Steuerung erzeugt.
*MEMR	A	B12	Speicher-Lesebefehl. Dieses Signal veranlaßt den Speicher des PC, Daten auf den Datenbus zu legen. Die Ansteuerung erfolgt wie bei MEMW.
*IOW	A	B13	Steckkarten-Schreibbefehl. Durch diesen Befehl weist der Computer eine Karte an, Daten vom Datenbus zu lesen. Die Ansteuerung erfolgt wie bei MEMW.
*IOR	A	B14	Steckkarten-Lesebefehl. Über diese Leitung wird eine Karte angewiesen, Daten auf den Datenbus zu legen. Die Ansteuerung erfolgt wie bei MEMW.
*DACK3	A	B15	DMA-Bestätigung. Dieses Signal quittiert die über das DRQ-Signal gestellte DMA-Anforderung.
DRQ3	E	B16	Siehe DRQ2
*DACK1	A	B17	Siehe DACK3
DRQ1	E	B18	Siehe DRQ2
*DACK0/ Refresh	A	B19	DMA-Bestätigung. Diese Leitung signalisiert einen Refreshzyklus.
CLK	A	B20	Systemtakt des Computers. Bei 4,77 MHz hat er eine Zykluszeit von 210ns.
IRQ7	E	B21	Siehe IRQ2
IRQ6	E	B22	Siehe IRQ2
IRQ5	E	B23	Siehe IRQ2
IRQ4	E	B24	Siehe IRQ2
IRQ3	E	B25	Siehe IRQ2
*DACK2	A	B26	Siehe DACK3
T/C	A	B27	Terminalzähler. Dieses Signal liefert einen Impuls, wenn der Terminalzählerstand für einen beliebigen DMA-Kanal erreicht ist.
ALE	A	B28	Adressenverriegelung. Dieser Anschluß wird von der Systemplatine verwendet, um die derzeit gültige Adresse gegenüber dem Prozessor zu verriegeln. Die Verriegelung erfolgt durch den Bus-Controller 8288 (PC) beziehungsweise 82288 (AT). In Verbindung mit dem Signal AEN steht ALE auf dem Steckplatzbus als Hinweis für eine gültige Prozessoradresse zur Verfügung.
+5V	–	B29	Positive Versorgungsspannung vom Netzteil.
OSC	A	B30	Oszillatortakt. An diesem Anschluß läßt sich die Systemquarz-Frequenz von 14,318 MHz direkt abgreifen.
GND	–	B31	Masse. Dieser Anschluß dient als Bezugspotential für alle Signale und Versorgungsspannungen.

* Alle Signale mit einem Stern vor der Bezeichnung arbeiten mit negativer Logik, das heißt, sie sind nullaktiv.

** Bei einem AT führt der Kontakt B4 nicht direkt auf den Eingang IRQ2 des ersten Interrupt-Controller, sondern auf IRQ9 am zweiten Interrupt-Controller. Der entsprechende Interrupt wird per Software entsprechend umgeleitet. Dieses Verfahren ist notwendig, weil ein AT mit zwei Interrupt-Controllern ausgestattet ist, die kaskadiert, also hintereinandergeschaltet sind. Der zweite Interrupt-Controller, der die Leitungen IRQ8 bis IRQ15 bearbeitet, hängt am Eingang IRQ2 des ersten Controller.

Tabelle 1. Diese Signale stehen am Steckplatz eines PC für die Erweiterungskarten zur Verfügung

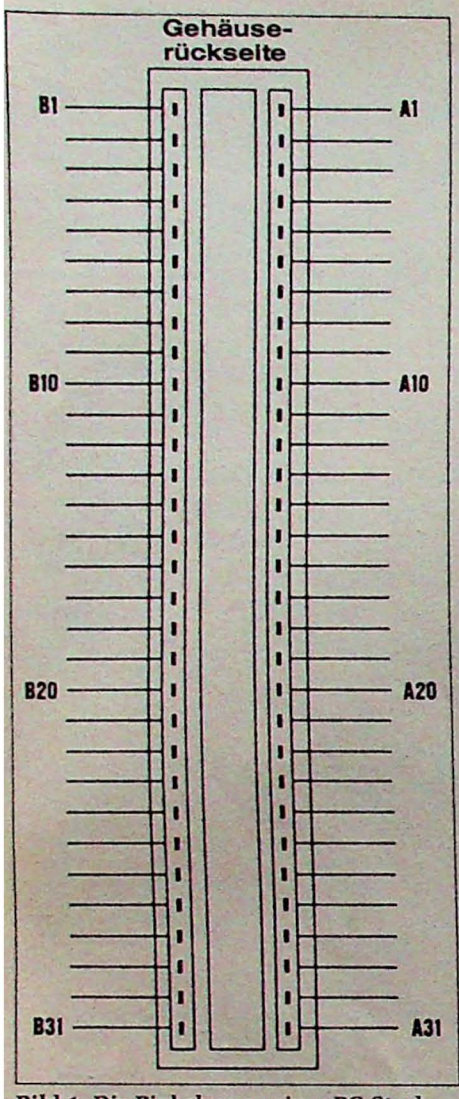


Bild 1. Die Pinbelegung eines PC-Steckplatzes

der PC-Mutterplatine für jeden Steckplatz eine 62polige Steckverbindung eingelötet, die den entsprechenden Platinenstecker der Erweiterungskarte aufnimmt.

An einem Steckplatz steht der gesamte Bus des PC zur Verfügung. (Unter einem Bus versteht man im Computerbereich eine Ansammlung aller wichtigen Signale des Computers, über die der Prozessor mit der Peripherie kommuniziert). Die Signale sind an jedem Steckplatz identisch, und alle Steckplätze sind untereinander gleichberechtigt.

Alle Signalleitungen an den PC-Steckplätzen führen TTL-Pegel (+ 5 Volt oder 0 Volt), es gibt keine Analogleitungen wie zum Beispiel beim Mikrokanal der neuen PS/2-Rechner. Jeder Steckplatz stellt alle 20 Adreßleitungen A0 bis A19, die acht Datenleitungen D0 bis D7, sechs Interrupt-Signale, sechs DMA-Signale sowie alle für die Speicherverwaltung und Adressierung notwendigen Signale zur Verfügung.Auch Taktsignale, Spannungsleitungen und einige weitere Steuersignale lassen sich an jedem Steckplatz abgreifen.

Damit die einzelnen Signalleitungen der Bausteine auf der Mutterplatine nicht durch die Erweiterungskarten überlastet werden, sind alle Leitungen durch Treiberbausteine gepuffert. Daher dürfen auch mehrere Steckkarten eine Signalleitung gleichzeitig belasten, ohne daß der entsprechende Baustein auf der Mutterplatine Schaden nimmt.

Die Leitungstreiber sind so ausgelegt, daß pro Steckplatz und Leitung zwei TTL-Lasteinheiten angeschlossen werden dürfen. Das bedeutet, daß jede Leitung eines Steckplatzes in der Lage ist, ohne Schaden zwei Gatter eines TTL-Bausteins zu steuern.

Wenn Sie eigene Erweiterungskarten entwickeln möchten, ist diese Information für Sie sehr nützlich, um eine Beschädigung der Bausteine auf der Mutterplatine zu vermeiden.

An die Leitungen eines Steckplatzes lassen sich jeweils zwei TTL-Lasteinheiten anschließen

Wollen Sie mehr als zwei TTL-Gatter aus einer Signalleitung speisen, so müssen Sie eigene Treiberbausteine zwischenschalten. Handelsübliche Erweiterungskarten für PCs belasten die Signale jedoch gewöhnlich nur mit je einer TTL-Last, so daß keine Gefahr der Überlastung besteht. Außerdem führen bestimmte Kontakte die vier Betriebsspannungen von + 12 Volt, + 5 Volt, -5 Volt und -12 Volt. Zur Stromversorgung darf eine Erweiterungskarte an diesen Kontakten in begrenztem Umfang Strom entnehmen.

Wenn eine Erweiterungskarte die sofortige Aufmerksamkeit des Prozessors benötigt, dann fordert sie diese über eine der sechs Interrupt-Leitungen (IRQ2 bis IRQ7) an. Die Interrupt-Leitungen sind mit dem Interrupt-Controller 8259 verbunden, der gleichzeitig eintreffende Interrupt-Anforderungen ihrer Priorität nach ordnet und an den Prozessor weiterleitet.

Mit der Datenübertragung von der Steckkarte auf die Mutterplatine und umgekehrt muß nicht unbedingt der Prozessor beschäftigt werden, denn für diese Aufgabe enthält ein PC einen und ein AT zwei DMA-Controller. Um einen solchen DMA-Transfer anzufordern, legt die entsprechende Erweiterungskarte eine der drei Leitungen DRQ1 bis DRQ3 auf 1. Eine Transfer-Anforderung beantwortet der DMA-Controller 8237 über eine der drei DMA-Bestätigungssignale DACK1 bis DACK3.

Tabelle 1 zeigt Ihnen detailliert die Bedeutung der Signale an einem PC-Steckplatz, während aus Bild 1 die genaue Pinbelegung der Steckverbindung für den Platinenstecker einer Erweiterungskarte hervorgeht.

Über die Interrupt-Leitungen können Erweiterungskarten die Aufmerksamkeit des Prozessors anfordern

Für bestimmte Steckkarten ist die Synchronisation mit dem Computer wichtig. Aus diesem Grund ist sowohl der Systemtakt, der sich allerdings von Modell zu Modell unterscheidet, als auch der Takt des 14,318-MHz-Systemquarzes an jedem Steckplatz abgreifbar.

Falls eine langsame Erweiterungskarte mit der Systemgeschwindigkeit nicht mitkommt, kann sie unter Verwendung der Leitung »I/O CK RDY« (zu deutsch etwa »Ein-/Ausgabe-Steckplatz bereit«) die Mutterplatine dazu bewegen, die Port- beziehungsweise Speicherzugriffszyklen zu verlängern. Dabei werden die Maschinenzyklen immer um ein Vielfaches eines Taktzyklus ausgedehnt. Diese Leitung darf nicht länger als 2,5 Mikrosekunden aktiv sein.

Weil der Arbeitsspeicher eines PC regelmäßig aufgefrischt werden muß (dieser Vorgang wird als »Refresh« bezeichnet), löst ein Zeitgeberbaustein in regelmäßigen Abständen (bei den meisten Rechnern alle 15 Mikrosekunden) einen Refresh-Zyklus für

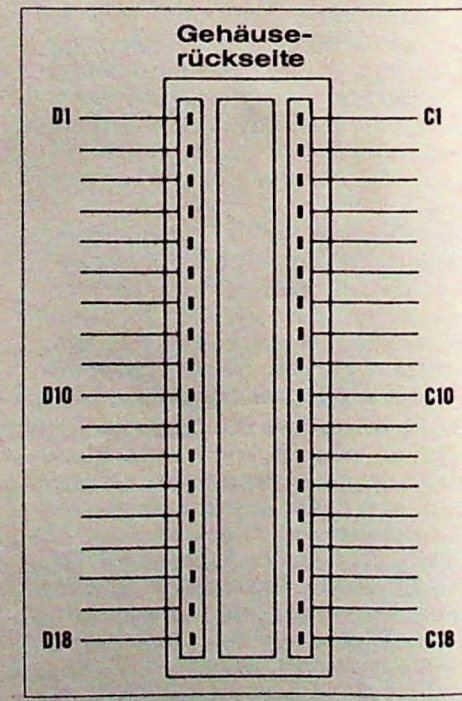


Bild 2. Der 36polige AT-Stecker enthält die zusätzlichen, AT-spezifischen Signalleitungen

die dynamischen Speicherbausteine aus. Das Signal DACK0 (bei einem PC) beziehungsweise »Refresh« (bei einem AT) am Anschluß B19 signalisiert einen Refresh-Zyklus.
Um Portadressen auf den Erweiterungskarten anzusprechen, stehen bei einem PC 1024 Adressen zur Verfügung, das heißt, für die Portadressierung werden nur die Adreßleitungen A0 bis A9 berücksichtigt.
Wenn Paritätsfehler im Arbeitsspeicher oder bei Portzugriffen auf einer Steckkarte auftreten, kann die Steckkarte über die Leitung »I/O CH CHK« den nichtsperrbaren Interrupt NMI auslösen, der eine entsprechende Meldung für den Benutzer ausgibt und die Programmausführung stoppt. Die Auslösung eines NMI durch Steckkarten läßt sich ebenso wie auf der Mutterplatine softwareseitig sperren.
Bis jetzt wurden noch keine Unterschiede zwischen PC- und AT-Steckplätzen erkennbar, doch es gibt sie: Bei der Entwicklung des IBM AT hat IBM auf Abwärtskompatibiltät geachtet, das heißt, die Software und die Erweiterungskarten des IBM PC sollten auch für den AT geeignet sein. Das ist recht gut gelungen, wenn auch gewisse Abstriche in Sachen Kompatibilität hingenommen wurden, um Innovationen einzuführen. Aus diesem Grund besteht ein AT-Steckplatz aus zwei Steckverbindungen, der 62poligen-PC-Verbindung und einem zusätzlichen 36poligen Steckkontakt, an dem die neu hinzugekommenen Signale abgreifbar sind. Die Bedeutung der Signale des AT-spezifischen Steckverbinders entnehmen Sie Tabelle 2, und die Pinbelegung der AT-Steckverbindung sehen Sie in Bild 2.
Bedenken Sie dabei, daß wegen der etwas unterschiedlichen Hardware von PC und AT nicht alle PC-Karten im AT funktionieren, auch wenn am Steckplatz die gleichen Signale anliegen. Am PC-kompatiblen Steckplatz haben sich nur drei Leitungen geändert, wobei die Änderungen aber in der Regel keine Bedeutung haben.
Zunächst wurde die Leitung IRQ2 (B4) durch IRQ9 ersetzt. Bei einem AT führt der Kontakt B4 nicht direkt auf den Eingang IRQ2 des ersten Interrupt-Controllers, sondern auf IRQ9 am zweiten Interrupt-Controller. Der entsprechende Interrupt wird per Software entsprechend umgeleitet. Dieses Verfahren ist notwendig, weil ein AT mit zwei Interrupt-Controllern ausgestattet ist, die kaskadiert, also hintereinandergeschaltet sind. Der zweite Interrupt-Controller, der die Leitungen IRQ8 bis IRQ15 bearbeitet, hängt am Eingang IRQ2 des ersten Controllers. Die Leitung DACK0 führt nicht mehr die Quittierung des Kanal 0 des DMA-Controllers, sondern ein adäquates Signal der (beim AT anders gearteten) Refresh-Logik. Für eine Erweiterungskarte spielt dies jedoch keine Rolle, das Verfahren beim AT hat nur den Vorteil, daß sich der Kanal 0 des (ersten) DMA-Controller, der beim PC den Refresh übernimmt, beim AT für andere Aufgaben verwenden läßt.
Die Leitung »Card Select«, die nur bei PCs und dort nur am achten Steckplatz vorhanden war (und meist nicht verwendet wurde), ist durch das Signal »0WS« ersetzt worden. Dieses Signal gibt bekannt, daß der derzeitige Datentransfer ohne zusätzliche Wartezyklen (»Waitstates«) ausgeführt werden kann.
An der AT-spezifischen Steckverbindung liegen außerdem die Adreßleitungen für die Adressierung des 16 MByte großen Speicherbereichs im Protected Mode sowie die Datenleitungen D8 bis D15 für 16-Bit-Transfers an. Über die Leitungen D1 »MEM CS16« beziehungsweise D2 »I/O CS16« bestimmt eine Erweiterungskarte, daß der Datentransfer auf Portadresse oder in den Arbeitsspeicher 16-Bit-weise erfolgt.

Für 386er PCs gibt es – im Gegensatz zum AT – keine standardisierte Steckplatzerweiterung, weil hier kein Vorbild von IBM existiert. Im September letzten Jahres hat jedoch eine Reihe namhafter Computerhersteller unter Federführung von Compaq die Entwicklung eines einheitlichen Zusatzbusses für 386er PCs unter der Bezeichnung »Eisa« bekanntgegeben, der als Reaktion auf IBMs Mikrokanal zu sehen ist.

Der Fortentwicklung des »alten« PC-Bus setzen allerdings physikalische Gegebenheiten (wie zum Beispiel die Störstrahlung) früher oder später Grenzen, so daß der Eisa-Bus seine (von den Herstellern versprochene) Leistungsfähigkeit erst noch beweisen muß.

(Martin Hepp/ma)

Signal	E = Eingang A = Ausgang	Pin	Signalbeschreibung
SBHE	E/A	C1	»Bus High Enable«. Dieser Anschluß signalisiert, daß ein Datentransfer auf der oberen Hälfte des Adreßbus stattfindet.
LA23 bis LA17	E/A	C2 bis C8	Adreßleitungen A23 bis A17. Über diese zusätzlichen Adreßleitungen sprechen der Prozessor und DMA-Controller den erweiterten Speicher im Protected Mode an.
*HMEMR	E/A	C9	Speicher-Lesebefehl. Dieses Signal veranlaßt den Arbeitsspeicher, die Daten der adressierten Speicherzelle auf den Datenbus zu legen. Die Leitung MEMR des einfachen PC-Steckplatzes wird nur dann verwendet, wenn sich die adressierte Speicherzelle innerhalb des 1 MByte umfassenden Speichers im Real Mode des 80286 befindet.
*HMEMW	E/A	C10	Speicher-Schreibbefehl. Dieses Signal veranlaßt den Arbeitsspeicher, die auf dem Datenbus anliegenden Daten zu speichern. Die Leitung MEMW des einfachen PC-Steckplatzes wird nur dann verwendet, wenn sich die adressierte Speicherzelle innerhalb des 1 MByte umfassenden Speichers im Real Mode des 80286 befindet.
SD08 bis SD15	E/A	C11 bis C18	Datenleitungen 8 bis 15. Diese zusätzlichen Anschlüsse liefern bei 16-Bit-Transfers die Datenbit 8 bis 15.
*MEMCS16	E	D1	16-Bit-Speicherzugriff. Diese Leitung zeigt an, daß es sich bei dem derzeitigen Datentransfer um einen 16-Bit-Speicherzugriff handelt.
*I/O_CS16	E	D2	16-Bit-Portzugriff. Diese Leitung zeigt an, daß es sich bei dem derzeitigen Datentransfer um einen 16-Bit-Portzugriff handelt.
IRQ10	E	D3	Interrupt-Anforderung einer Karte. Bei einem AT haben die Interrupts IRQ8 bis IRQ15 die höchste Priorität nach IRQ0 (Timer) und IRQ1 (Tastatur). Erst dann werden IRQ3 bis IRQ7 bearbeitet. Die Leitungen IRQ8 und IRQ9 sind auf der AT-Mutterplatine fest belegt. Bei einer Interrupt-Anforderung geht der Anschluß auf 1 und verbleibt bis zur Quittierung durch den Prozessor (Acknowledge) in diesem Zustand.
IRQ11	E	D4	Siehe IRQ10
IRQ12	E	D5	Siehe IRQ10
IRQ15	E	D6	Siehe IRQ10
IRQ14	E	D7	Siehe IRQ10
*DACK0	A	D8	DMA-Bestätigung. Dieses Signal quittiert die über das DRQ-Signal ausgelöste DMA-Anforderung.
DRQ0	E	D9	DMA-Anforderung. Asynchrone Anforderung eines Peripheriegeräts für den Zugriff auf eine DMA-Funktion. DRQ0 hat die höchste und DRQ7 die niedrigste Priorität. Die Leitungen DRQ0 bis DRQ3 lösen die Ausführung eines 8-Bit-DMA-Transfers aus, die Leitungen DRQ5 bis DRQ7 lösen einen 16-Bit-DMA-Transfer aus. DRQ4 ist auf der AT-Mutterplatine fest belegt und steht daher an den Steckplätzen nicht zur Verfügung. Bei einer Anforderung geht die Leitung auf 1 und verbleibt bis zur Aktivierung der korrespondierenden DACK-Leitung in diesem Zustand.
*DACK5	A	D10	Siehe DACK0
DRQ5	E	D11	Siehe DRQ0
*DACK6	A	D12	Siehe DACK0
DRQ6	E	D13	Siehe DRQ0
*DACK7	A	D14	Siehe DACK0
DRQ7	E	D15	Siehe DRQ0
+5V	–	D16	Positive Versorgungsspannung vom Netzteil.
*MASTER	E	D17	Über diese Leitung kann ein Zusatzprozessor oder DMA-Controller die völlige Kontrolle über das Computersystem erhalten. Wenn MASTER auf 0 gesetzt ist, kann der Zusatzprozessor auf alle Daten-, Adreß- und Steuerleitungen zugreifen. Dieser Anschluß darf jedoch nicht länger als 15 Mikrosekunden auf 0 gesetzt sein, weil sonst der Arbeitsspeicher seinen Inhalt aufgrund des fehlenden Refresh verliert.
GND	–	D18	Masse. Dieser Anschluß dient als Bezugspotential für alle Signale und Versorgungsspannungen.

Tabelle 2. An der zusätzlichen 36poligen Steckverbindung eines AT-Steckplatzes lassen sich die speziellen Signale des AT abgreifen

Der DMA-Controller 8237

In einem Computersystem müssen sehr häufig Daten zwischen Arbeitsspeicher und Peripherie-Einheiten übertragen werden. Dazu ist im PC ein DMA-Controller (DMA für »Direct Memory Access«) vorgesehen, der die Datenübertragung wesentlich schneller als der Prozessor ausführt.

Ein DMA-Controller (im PC wird der Typ 8237 verwendet) ist zu nicht viel mehr in der Lage, als stupide Daten »hin- und herzuschaufeln«. Doch gerade diese Spezialisierung auf eine einzige Aufgabe erlaubt es, daß der DMA-Controller diese Arbeit bis zu zehnmal schneller erledigt als der Prozessor des PC.
Wenn eine Peripherie-Einheit wie zum Beispiel der Disketten-Controller Daten in den Arbeitsspeicher des PC übertragen möchte, dann aktiviert sie eine der DREQ-Leitungen (DMA-Anforderung) des DMA-Controllers. Dieser bestätigt die Anforderung einer DMA – Übertragung über die dazugehörende DACK-Leitung (DMA-Bestätigung).
Darauf koppelt der DMA-Controller den Prozessor von den Peripherie-Einheiten und vom Arbeitsspeicher ab und bedient die Signalleitungen des Speichers und der Peripherie so, wie bei einem Zugriff des Prozessors. Der DMA-Controller fordert also beispielsweise eine Peripherie-Einheit auf, ein Datenbyte auf den Datenbus zu legen und weist eine Adresse des Arbeitsspeichers an, die auf dem Datenbus liegenden Daten zu übernehmen.
Eine solche Übertragung erledigt der DMA-Controller in deutlich kürzerer Zeit als der Prozessor. Ein kompletter Zyklus dauert nur fünf Takte, während der Prozessor dafür 29 Takte benötigen würde. Weil der 8237 vier getrennte Kanäle besitzt, ist er zudem in der Lage, vier solche Datenübertragungen gleichzeitig durchzuführen.
Überträgt der DMA-Controller Daten zu einem Peripheriegerät, so aktiviert er die Leitungen MEMWR, IOW und DACK.
Da sich, dadurch und durch die zusätzlich anliegende Adresse bedingt, auch andere Peripherie-Einheiten angesprochen fühlen können, signalisiert der DMA-Controller über das Signal AEN, daß die derzeite Adressierung nicht vom Prozessor stammt.
Für eine gewöhnliche Datenübertragung, von einer Peripherie-Einheit in den Arbeitsspeicher oder umgekehrt, ist jeweils nur ein DMA-Kanal notwendig. Anders sieht es dagegen bei einer Verschiebung innerhalb des Speichers aus. Hierzu benötigt der 8237 zwei DMA-Kanäle. Da es in diesem Fall keine Peripherie-Einheit gibt, die die DMA-Anforderung über eine DREQ-Leitung stellt, muß die Auslösung per Software erfolgen. Dazu dient das Kanalanforderungsregister, dessen Belegung Sie der Tabelle 1 entnehmen.

Bei einer solchen, »Memory to Memory« genannten, Datenübertragung übernimmt Kanal 0 die Quelladresse und Kanal 1 die Zieladresse. Genaugenommen liest Kanal 0 ein Byte aus dem Speicher, speichert es in einem temporären Register und Kanal 1 schreibt das Byte an die Zieladresse.

Leider ist bei einem PC eine solche DMA – Übertragung innerhalb des Arbeitsspeichers nicht möglich, denn Kanal 0 übernimmt die regelmäßige Wiederauffrischung des Arbeitsspeichers und darf daher nicht für andere Zwecke verwendet werden. Es ist ferner zu beachten, daß der Prozessor während einer DMA – Übertragung kein Programm ausführen kann und die Beendigung der Übertragung abwarten muß.

Die Programmierung des DMA-Controllers erfolgt mit Hilfe zahlreicher Befehle und Adressen. Zur Steuerung eines jeden einzelnen Kanals steht dem Programmierer ein sechs Bit breites Modusregister zur Verfügung, das über die Portadresse 00Bhex angesprochen wird. Tabelle 2 zeigt den Aufbau des Modusregisters.

Das Kanalanforderungsregister	
Zustand	**Bedeutung**
Bit 0, Bit 1 = 00	Anforderung von Kanal 0
Bit 0, Bit 1 = 01	Anforderung von Kanal 1
Bit 0, Bit 1 = 10	Anforderung von Kanal 2
Bit 0, Bit 1 = 11	Anforderung von Kanal 3
Bit 2 = 0	Anforderungsbit löschen
Bit 2 = 1	Anforderungsbit setzen

Tabelle 1. Der Aufbau des Kanalanforderungsregisters

Das Modusregister	
Zustand	**Bedeutung**
Bit 0, Bit 1 = 00	Kanal 0 angewählt
Bit 0, Bit 1 = 11	Kanal 1 angewählt
Bit 0, Bit 1 = 10	Kanal 2 angewählt
Bit 0, Bit 1 = 11	Kanal 3 angewählt
Bit 2, Bit 3 = 01	Verify-Übertragung
Bit 2, Bit 3 = 01	Schreib-Übertragung
Bit 2, Bit 3 = 10	Lese-Übertragung
Bit 4 = 1	Auto-Initialisierung einschalten
Bit 5 = 0	Adressen jeweils um 1 erhöhen
Bit 5 = 1	Adressen jeweils um 1 vermindern
Bit 6, Bit 7 = 00	Anforderungsmodus
Bit 6, Bit 7 = 01	Einzelübertragung
Bit 6, Bit 7 = 10	Blockübertragung
Bit 6, Bit 7 = 11	Kaskadierungsmodus

Tabelle 2. Der Aufbau des Modusregisters

Das Kanalmaskierungsregister	
Zustand	**Bedeutung**
Bit 0, Bit 1 = 00	Maskierung für Kanal 0
Bit 0, Bit 1 = 01	Maskierung für Kanal 1
Bit 0, Bit 1 = 10	Maskierung für Kanal 2
Bit 0, Bit 1 = 11	Maskierung für Kanal 3
Bit 2 = 0	Maskierungsbit löschen
Bit 2 = 1	Maskierungsbit setzen

Tabelle 3. Der Aufbau des Kanalmaskierungsregisters

Das Hauptsteuerregister	
Zustand	**Bedeutung**
Bit 0 = 1	Speicher-zu-Speicher – Übertragung ein
Bit 1 = 1	Adresse von Kanal 0 speichern
Bit 2 = 1	Controller sperren
Bit 3 = 1	Komprimierte Zeitabfrage
Bit 4 = 1	Rotierende Priorität
Bit 5 = 1	Schreibbefehlserweiterung
Bit 6 = 0	Signal DREQ ist aktiv 0
Bit 6 = 1	Signal DREQ ist aktiv 1
Bit 7 = 0	Signal DACK ist aktiv 0
Bit 7 = 1	Signal DACK ist aktiv 1

Tabelle 4. Der Aufbau des Hauptsteuerregisters

Abhängig von Bit 5 inkrementiert oder dekrementiert der DMA-Controller die aktuelle Adresse nach der Übertragung eines Byte und dekrementiert gleichzeitig den aktuellen Zähler. Ist dieser Zähler auf Null gestellt oder die Leitung EOP aktiviert, so endet die Übertragung. Dann erhalten die Basisregister die Anfangswerte der aktuellen Register, so daß sie bereit sind, eine neue Information aufzunehmen. Das ist der Fall, sobald der Zähler auf Null steht und Bit 4 den Wert 1 hat.

Über die beiden Bits 2 und 3 geben Sie entweder einen Schreib- oder Lesebefehl, und mit den Bit 6 und 7 stellen Sie die verschiedenen Übertragungsarten wie Anforderungs-, Einzel-, Block- oder Kaskadierungsmodus ein.

In der Praxis hat es sich als günstig erwiesen, bei Übertragungsgeschwindigkeiten von weniger als einem Byte pro vier Taktzyklen im Einzelübertragungsmodus zu arbeiten, denn hierbei überträgt das System mit jeder Aktivierung des DREQ-Signals nur ein Byte. Dieser Modus hat den Vorteil, daß dem Prozessor zwischen den beiden Zyklen genug Zeit bleibt, um andere, wichtige Aufgaben zu erledigen.

Ist eine angeschlossene Peripherie-Einheit dagegen zu einer höheren Übertragungsrate in der Lage, dann ist es sinnvoll, den Blockübertragungsmodus des DMA-Controllers zu wählen. In diesem Modus überträgt der DMA-Controller so viele Bytes hintereinander, bis der interne Zähler auf Null herabgezählt ist.

Der DMA-Controller besitzt mit dem Kanalmaskierungs- und dem Hauptsteuerregister noch zwei weitere Kontrollregister. Mit dem Kanalmaskierungsregister lassen sich alle vier DMA-Kanäle einzeln dazu bewegen, die ihnen zugewiesene Leitung DREQ zu ignorieren. Die Portadresse für das Kanalmaskierungsregister lautet 00Ahex. Tabelle 3 zeigt den Aufbau dieses Registers. Wie schon beim Kanalanforderungsregister, so erfolgt auch beim Kanalmaskierungsregister die Manipulierung der einzelnen Kanäle mit Hilfe der ersten beiden Bit.

Das letzte wichtige Kontrollregister des DMA-Controllers ist das Hauptsteuerregister, das Sie über die Portadresse 008hex ansprechen. In diesem Register legen Sie die Betriebsarten des Controller fest und nehmen damit die Grobsteuerung für die anderen schon beschriebenen Befehlsregister vor. Tabelle 4 zeigt den Aufbau des Hauptsteuerregisters.

Das erste Bit dient zur Freigabe der Speicher-zu-Speicher-Übertragung. Beachten Sie, daß der DMA-Controller maximal nur 64 KByte große Blöcke auf einmal verschieben kann. Mit dem zweiten Bit können sie den Controller dazu veranlassen, die aktuelle Adresse des ersten Übertragungskanals zu speichern und festzuhalten.

Der DMA-Controller 8237 verschiebt nur maximal 64 KByte große Datenblöcke

Setzen Sie das Bit 2, so ist der Controller so lange gesperrt, bis sich der Zustand dieses Bit wieder ändert. Damit vermeiden Sie, daß dem Prozessor während der Ausführung besonders wichtiger Aufgaben die Kontrolle entzogen wird, um einen DMA-Zyklus auszuführen. Weil beim PC der Refresh des Arbeitsspeichers über den DMA-Controller ausgeführt wird, darf der Controller nicht von einem Programm gesperrt werden, sonst sind Datenverluste und ein damit verbundener »Parity-Error« denkbar. Bit 3 des Hauptsteuerregisters gestattet eine häufigere Abfrage der einzelnen DMA-Kanäle, was jedoch bei Diskettenzugriffen zu Zeitproblemen führen kann. Die rotierende Priorität, die mit Bit 4 bestimmt wird, erlaubt den schnellen Wechsel einzelner Prioritätsstufen, was bei Interrupt-Behandlungen dienlich ist.

Weil der DMA-Controller vom Typ 8237 noch aus der Zeit des 8080--Prozessors stammt, unterstützt er nur einen 64 KByte großen Adreßbereich. Daher lassen sich – wie bereits erwähnt – nur maximal 64 KByte Daten an einem Stück übertragen. Ein PC verfügt aber über einen Adreßraum von einem MByte. Die fehlenden 4 Bit werden deshalb über ein sogenanntes DMA-Seitenregister festgelegt. Im PC existiert für jeden Kanal des DMA-Controller ein solches Seitenregister. Die vom DMA-Controller angesprochene Speicheradresse setzt sich folglich aus dem Inhalt des Seitenregisters und den 16 Adreßleitungen des DMA-Controllers zusammen.

Im Gegensatz zu einem PC enthält ein AT zwei DMA-Controller, die hintereinandergeschaltet sind. Dies nennt man auch Kaskadierung. Aus diesem Grund enthält der AT nicht vier, sondern sieben DMA-Kanäle. Dabei sind die Kanäle 0 bis 3 des ersten DMA-Controller für 8-Bit-DMA-Übertragungen konzipiert und die Kanäle 4 bis 7 für 16-Bit-Übertragungen. Die Seitenregister umfassen beim AT 8 Bit, damit der DMA-Controller auf den vollständigen, 16 MByte großen Adreßbereich des im AT verwendeten 80286-Prozessors zugreifen kann. Das bedeutet, der DMA-Controller kann, auch wenn sich der 80286-Prozessor im 8086-kompatiblen Real-Modus befindet, auf den erweiterten Arbeitsspeicher (»Extended Memory«) zugreifen.

Leider ist beim AT keine sinnvolle Speicher-zu-Speicher-Übertragung programmierbar, obwohl der Refresh-Vorgang hier nicht mehr von Kanal 0 des DMA-Controllers, sondern von einem speziellen, höherprivilegierten Baustein erledigt wird.

(Martin Hepp/ma)

Wer — wo — wann?

Bei Zuschriften bitten wir, folgendes zu beachten: Haben Sie Fragen, Anregungen oder Programme, die Sie der DOS-Redaktion zukommen lassen möchten, adressieren Sie wie folgt:

Redaktionsbüro Everts & Hagedorn · Redaktion DOS International
Hans-Stießberger-Str. 3 · 8013 Haar/München · Tel. (089) 46 10 01-0

Haben Sie Fragen, die zum Beispiel Abonnements, Kleinanzeigen oder Databox-Bestellungen betreffen, wenden Sie sich bitte an den Verlag:

DMV-Verlag · Postfach 250 · 3440 Eschwege

Der Interrupt-Controller 8259

Wenn ein Baustein im PC die sofortige Aufmerksamkeit des Prozessors benötigt, löst er einen Interrupt aus, der den Prozessor dazu veranlaßt, die Ausführung des laufenden Programms zu unterbrechen und eine bestimmte Serviceroutine aufzurufen. Der Interrupt-Controller 8259 sorgt dafür, daß die verschiedenen Interrupts in der richtigen Reihenfolge bearbeitet werden.

Der Prozessor des PC kann jeweils nur eine bestimmte Aufgabe zur gleichen Zeit ausführen, also beispielsweise ein Anwendungsprogramm bearbeiten oder einen Peripheriebaustein abfragen. Wenn nun ein Zeichen von der seriellen Schnittstelle empfangen wurde und der Prozessor dieses sofort lesen und auswerten muß, dann ist es notwendig, daß der Prozessor die Ausführung des laufenden Programms umgehend unterbricht und sich um den seriellen Schnittstellenbaustein kümmert. Die Anforderung für diese Unterbrechung erzeugt der Schnittstellenbaustein selbst.

Eine entsprechende Unterbrechungsroutine verarbeitet dann das eingelesene Zeichen entweder direkt oder speichert es in einem Puffer zwischen. Dieser Vorgang muß sofort erfolgen, denn das nächste empfangene Zeichen überschreibt das vorige, weil der Schnittstellenbaustein keinen eigenen Zwischenspeicher enthält.

Das Verfahren der gezielten Unterbrechungen hat zur Folge, daß der Prozessor die angeschlossene Peripherie nicht ständig abfragen muß und sich ganz der Bearbeitung des laufenden Programms widmen kann.

Eine Unterbrechnung des Prozessors wird auch als »Interrupt« bezeichnet. Um einen Interrupt auszulösen, muß der betreffende Peripheriebaustein die Leitung »INTR« des Prozessors auf + 5 Volt legen. Beim nächsten Taktzyklus reagiert der Prozessor auf diese Interrupt-Anforderung, indem er den Bus-Controller 8288 (beim AT der Typ 82288) veranlaßt, über die Leitung »INTA« (»Interrupt Acknowledge«) die Annahme des Interrupt zu bestätigen.

Der Interrupt-Controller vom Typ 8259 legt dann die Nummer des Interrupt-Vektors, der die Adresse der aufzurufenden Bearbeitungsroutine enthält, auf den Datenbus. Der Prozessor multipliziert diesen Wert mit vier und liest unter der Adresse, die sich aus dieser Multiplikation ergibt, einen vier Byte langen Interrupt-Vektor aus den ersten 1024 Byte des Arbeitsspeichers. Darauf sichert der Prozessor das Statusbit-Register auf den Stapel (Stack) und ruft über eine Intersegment-Adressierung (bestehend aus einer 16-Bit-Offset- und einer 16-Bit-Segmentadresse) die in dem Interrupt-Vektor angegebene Adresse auf. Hier befindet sich in aller Regel eine Interrupt-Serviceroutine, welche die Interrupt-Anforderung bedient und beispielsweise Zeichen an einen Drucker sendet oder Werte von der Tastatur übernimmt.

Ein Interrupt-Controller vom Typ 8259 verwaltet im PC die Unterbrechungsanforderungen der Hardware

Ein Peripheriebaustein fordert eine Unterbrechung des laufenden Programms über eine der Leitungen IRQ0 bis IRQ7 (IRQ bedeutet »Interrupt Request«) des Interrupt-Controller an. Dabei besitzt die Leitung IRQ0 die höchste Priorität, wird also beim gleichzeitigen Auftreten mehrerer Interrupt-Anforderungen als erste bearbeitet, und dann erst IRQ2, IRQ3 und so weiter. Die niedrigste Priorität ist IRQ7 zugeteilt. Nur wenn über keine andere Leitung eine Unterbrechung angefordert wird, bearbeitet der Interrupt-Controller dieses Signal sofort.

Treten mehrere Interrupts zugleich auf, so speichert der Interrupt-Controller diese im Interrupt-Anforderungs-Register und leitet zunächst die Interrupt-Anforderung mit der höchsten Priorität an den Prozessor weiter. Erst wenn die dazugehörende Routine den Befehl »Ende des Interrupt« (»EOI«) an den Controller schickt, bearbeitet dieser die folgenden, zwischengespeicherten Interrupt-Anforderungen.

Diese Verteilung von Prioritäten hat einen guten Grund, denn in einem Computersystem gibt es grundsätzlich Einheiten, welche die Aufmerksamkeit des Prozessors dringender benötigen als andere.

So ist beim PC der Zeitgeberbaustein mit IRQ0 verbunden und die Tastatur mit IRQ1. Daraus ergibt sich, daß der Prozessor zuerst die Systemuhr des Zeitgeberbausteins aktualisiert und erst dann die Tastatur ausliest. Die genaue Verwendung der acht IRQ-Leitungen des Interrupt-Controller entnehmen Sie der Tabelle 1. Tabelle 2 zeigt die zusätzlichen Interrupt-Leitungen eines AT.

In komplexen Computersystemen sind meist mehr als acht Interrupt-Leitungen notwendig. Zu diesem Zweck lassen sich bis zu acht Interrupt-Controller an einen weiteren solchen Controller anschließen.

Auf der Mutterplatine eines AT sind zwei Interrupt-Controller nach diesem Verfahren miteinander verbunden. Dabei übernimmt der erste Controller die Leitungen IRQ0 bis IRQ7, genau wie bei einem PC. Der zweite Controller bedient die zusätzlichen Interrupt-Leitungen IRQ8 bis IRQ15, die für die AT-spezifischen Bausteine zuständig sind. Dieser zweite Controller (»Slave« genannt) ist mit der IRQ-2-Leitung des ersten Controllers, des »Masters«, verbunden.

Wenn nun ein Peripheriebaustein eine Unterbrechung über den zweiten Interrupt-Controller anfordert, dann informiert dieser den ersten Controller darüber, daß er einen Interrupt bearbeiten muß. Der Master fordert nun die Aufmerksamkeit des Prozessors über die Leitung »INTR« an, was der Bus-Controller 8288 über »INTA« bestätigt. Nun wählt der Master über die drei Kaskadierungs-Leitungen CAS0 bis CAS2 den Interrupt-Con-

Interrupt	Bedeutung
IRQ0	Takt des zweiten Kanals des Zeitgeberbausteins
IRQ1	Tastatur-Interrupt
IRQ2	Grafikkarte
IRQ3	Zweite serielle Schnittstelle
IRQ4	Erste serielle Schnittstelle
IRQ5	Festplatten-Controller
IRQ6	Disketten-Controller
IRQ7	Parallele Schnittstelle

Tabelle 1. Die Hardware-Interrupts des PC

Interrupt	Bedeutung
IRQ8	Echtzeituhr
IRQ9	entspricht dem ursprünglichen IRQ2 in einem PC
IRQ10	reserviert
IRQ11	reserviert
IRQ12	reserviert
IRQ13	Arithmetikprozessor 80287
IRQ14	Festplatten-Controller
IRQ15	reserviert

Tabelle 2. Die zusätzlichen Hardware-Interrupts des AT

roller aus, der die Unterbrechung usgelöst hat. Dieser legt die Numner des aufzurufenden Interrupt-Vektors auf den Datenbus, und der Prozessor führt die entsprechende Routine aus.

Weil der zweite Interrupt-Controller in der Leitung IRQ2 des ersten angeschlossen ist, haben alle Interrupt-Leitungen des zweiten Controllers IRQ8 bis IRQ15) Vorrang vor dem zweiten Controller (IRQ3 bis IRQ7).

Für den Fall, daß der Prozessor gerade eine besonders wichtige, zeitkritische Aufgabe ausführt, läßt sich das Interrupt-Flag (»IF«) im Flag-Register mit dem Assemblerbefehl »cli« löschen. Dann ignoriert der Prozessor jede Interrupt-Anforderung, außer einem nichtmaskierbaren Interrupt, der direkt über die Leitung »NMI« am Prozessor ausgelöst wird.

Um nur bestimmte Interrupts zu sperren, enthält jeder Interrupt-Controller ein »Interrupt-Mask-Register«. Jedes Bit dieses Registers gehört zu einer der acht Interrupt-Leitungen. Bit 0 dieses Registers maskiert (bezieht sich auf) IRQ0, Bit 1 IRQ1 und so weiter.

Wenn Sie das entsprechende Bit im Interrupt-Mask-Register setzen, sperrt dies die Behandlung des betroffenen Interrupt, nicht aber dessen Speicherung. Nachdem die Sperrung eines Interrupt aufgehoben ist, leitet der Controller die zwischengespeicherten Interrupt-Anforderungen an den Prozessor weiter. Das Sperren eines höherprivilegierten Interrupt hat übrigens keinen Einfluß auf die übrigen Interrupts.

Das Interrupt-Mask-Register des ersten Interrupt-Controller wird über einen Schreibzugriff auf die Portadresse 21hex angesprochen, während das entsprechende Register des zweiten Controller (nur in einem AT) unter der Portadresse A1hex zu finden ist. Ein Lesezugriff auf dieses Register liefert den derzeitigen Status.

Neben dem Mask-Register gibt es noch ein weiteres wichtiges Register, über das der Programmierer Einfluß auf das Verhalten des Interrupt-Controllers ausüben kann. Dieses Interrupt-Befehlsregister wird über die Port-Adresse 20hex (erster Controller) beziehungsweise A0hex (zweiter Controller) adressiert.

Auf diese Weise läßt sich der Controller beispielsweise initialisieren oder einer der vier Betriebsmodi des 8259 einstellen. Die vollständigen Informationen können Sie bei Bedarf dem Datenbuch des Controller entnehmen.

Der Interrupt-Controller läßt sich über mehrere Register für seine Aufgaben programmieren

Die wichtigste Eigenschaft des Interrupt-Befehlsregisters ist, daß sich hier die Basisadresse der acht Interrupt-Vektoren festlegen läßt, die bei einer Unterbrechungsanforderung über IRQ0 bis IRQ7 (mit dem entsprechenden Offset 0 bis 28) auf den Datenbus gelegt wird. Die Bit 3 bis 7 dieses Wertes sind frei wählbar, während Bit 0 bis 2 der Nummer der Interrupt-Leitung entsprechen, die einen Interrupt angefordert hat.

In einem PC sind alle Interrupt-Eingänge des Interrupt-Controllers belegt. Der erste Eingang des Controllers ist mit dem Ausgang des zweiten Kanals des Zeitgeberbausteins 8253 verbunden. Der zweite Kanal des 8253 erzeugt 18,2mal pro Sekunde ein Taktsignal, das über den Anschluß an den Interrupt-Controller eine periodische Unterbrechungsanforderung stellt. Auf diese Weise entsteht das Taktsignal für die Systemuhr im PC.

Am zweiten Eingang des Interrupt-Controller ist die Tastatur angeschlossen. Sobald Sie auch nur eine einzige Taste kurz drücken, erzeugt die Tastatur eine Interrupt-Anforderung, um dem Prozessor mitzuteilen, daß eine Taste gedrückt wurde und diese ausgewertet werden muß.

Der dritte Eingang des 8259 ist für spezielle Interrupt-Anforderungen reserviert. Bei einem AT ist dort der zweite Interrupt-Controller angeschlossen.

Über den vierten Eingang des Interrupt-Controllers kann die zweite serielle Schnittstelle des PC – sofern vorhanden – eine Interrupt-Anforderung stellen und damit beispielsweise mitteilen, daß ein angeschlossenes Modem Daten senden möchte.

Der fünfte Eingang des Controllers ist für die erste serielle Schnittstelle vorgesehen. Auch diese kann über eine Interrupt-Anforderung anzeigen, daß Daten empfangen werden oder Sendebereitschaft besteht.

Der sechste und siebte Eingang des Interrupt-Controllers ist für den Anschluß eines Disketten- und Festplatten-Controllers vorgesehen, so daß auch diese Einheiten über eine Interrupt-Anforderung auf sich aufmerksam machen können.

Der achte und letzte Eingang ist schließlich für die parallele Druckerschnittstelle reserviert, die beispielsweise über eine Interrupt-Anforderung anzeigen kann, daß ein Übertragungsfehler aufgetreten ist.

Weil der Interrupt-Controller 8259 auch diejenigen Unterbrechungsanforderungen zwischenspeichert, die maskiert sind, haben die IBM-Ingenieure zusätzliche Ein/Ausgabe-Ports eingerichtet, über die sich die Peripheriebausteine vom Interrupt-Controller trennen lassen. Auf diese Weise läßt sich verhindern, daß die zwischenzeitlich angeforderten Unterbrechungen nicht bearbeitet werden, wenn die Maskierung aufgehoben wird.

(Martin Hepp/ma)

Toolboxen für Programmierer gesucht

Haben Sie eine Toolbox in C, Pascal, Basic, dBase, Clipper oder Assembler geschrieben? Beispielsweise schnelle, universelle Grafikroutinen, eine Textverarbeitungs- oder eine Datenbank-Toolbox? Dann sollten Sie sich schnellstmöglich mit uns in Verbindung setzen. Schicken Sie uns unverbindlich eine Diskette mit Ihrem Programm und einen Ausdruck der Dokumentation.

Falls Sie sich vorweg informieren wollen, zwischen 10.00 und 12.00 Uhr und zwischen 15.00 und 17.00 Uhr stehen wir Ihnen an allen Werktagen unter der Rufnummer (089) 4 61 00 10 zur Verfügung.

Schicken Sie Ihre Unterlagen an folgende Adresse:
Redaktionsbüro Everts & Hagedorn
»DOS Software«
Hans-Stießberger-Str. 3
8013 Haar/München

Der Zeitgeber 8253

Der Zeitgeber 8253 fungiert im PC als Mädchen für alles. Ob Uhrzeit, Speicher-Refresh oder Tonausgabe, überall ist der Zeitgeber beteiligt, so daß er trotz seiner vergleichsweise geringen Intelligenz (gegenüber den Controller-Bausteinen) unentbehrlich ist.

Zwar ist der Zeitgeber des PC im Prinzip zu nicht viel mehr fähig als zum Auf- und Abwärtszählen, doch dies immerhin in drei unabhängigen Kanälen. Und gerade diese einfachen Aufgaben sind für den PC von großer Wichtigkeit. Unterbleibt beispielsweise auch nur kurzzeitig der Refresh des Arbeitsspeichers, so »vergißt« der Speicher seinen Inhalt, und Sie müssen den Computer neu booten.

Der Zeitgeber 8253 ist ein 16-Bit-Zähler, der drei voneinander unabhängige Kanäle 0, 1 und 2 besitzt. Jeder dieser drei Kanäle ist in der Lage, einen Eingangstakt im Bereich von 0 bis 2 MHz zu übernehmen. Dieser Takt läßt sich durch einen beliebigen 16-Bit-Wert teilen, so daß am Ausgang eines Kanals praktisch jeder Takt bis zu einer Frequenz von 2 MHz erzeugt werden kann. Am Takteingang des Zeitgebers liegt im PC fest eine Frequenz von 1,193 MHz (4,77/4 MHz) an.

Der Ausgang des ersten Kanal 0 steht im PC für die Steuerung der internen Systemuhr zur Verfügung. Der Ausgang von Kanal 1 dient als Refresh-Signal für den DMA-Controller, um diesen zum Auffrischen der dynamischen Speicherbausteine zu veranlassen. Der Ausgang des Kanal 2 ist über ein Logikgatter mit dem eingebauten Lautsprecher verbunden, so daß sich dieser Kanal für die bescheidenen Sound-Fähigkeiten des PC verwenden läßt.

Im PC werden die Fähigkeiten des Zeitgebers nur zum Erzeugen von Ausgangssignalen benutzt, die dem durch eine 16-Bit-Zahl geteilten Wert des Systemtaktes entsprechen, obwohl jeder einzelne Kanal auch in der Lage ist, in einer von sechs möglichen Betriebsarten zu arbeiten, die im folgenden aufgeführt sind:

Modus 0: Erzeugen eines Interrupt, nachdem eine bestimmte Anzahl von Impulsen gezählt wurde oder nach einem Zählerüberlauf

Modus 1: Programmierbarer Monoflop

Modus 2: Taktgenerator

Modus 3: Rechtecksignalgenerator

Modus 4: Softwaremäßige Triggerung des Ausgangs

Modus 5: Hardwaremäßige Triggerung des Ausgangs

Im Modus 0 liegt die Leitung OUT (Ausgang) für eine bestimmte, programmierbare Anzahl von Impulsen, die über die CLK-Leitung (Takt) einlaufen, auf 0, um nach Erreichen des vorprogrammierten Zählerstandes wieder auf 1 zu gehen. Mit Hilfe dieser beiden Übergänge (erst von 1 auf 0 und anschließend wieder von 0 auf 1) läßt sich programmtechnisch die Arbeit des Prozessors für eine bestimmte Zeitdauer unterbrechen.

Für die drei Kanäle des Zeitgebers lassen sich sechs verschiedene Betriebsarten programmieren

Ähnlich arbeitet ein Kanal im Modus 1. Hierbei handelt es sich, wie auch im Modus 0, um die Programmierung eines Monoflops (monostabile Kippstufe). Allerdings startet hier die Zählfunktion nicht durch einen Befehl, sondern durch einen positiven Flankenwechsel an der Steuerleitung Gate (Tor). Man spricht in diesem Fall auch von einem zustandsgesteuerten Monoflop.

Modus 2 benutzt einen Divisor, der eine beliebige 16-Bit große Zahl teilt, um die Impulslänge festzulegen. Der Kanalausgang bleibt während der festgelegten Zeit auf 1 und geht nach Erreichen dieser Zeit für eine Taktperiode auf 0. Mit diesem Verhalten ergibt sich für den Ausgang ein asymmetrisches Taktverhältnis, dessen Periode der Eingangstakt, geteilt durch den 16-Bit-Wert, bestimmt.

Eine praktische Anwendung für diese Arbeitsweise, in Verbindung mit dem Kanal 1, ist das Erzeugen der Refresh-Zyklen für die dynamischen Speicherbausteine im PC.

Möchten Sie ein symmetrisches Signal am Ausgang eines Zählers erhalten, so müssen Sie den Modus 3 verwenden. In diesem Modus läßt sich ein symmetrischer, durch den 16-Bit-Wert geteilter Zähler programmieren. Sein Kennzeichen ist ein Ausgangssignal, dessen 0-Pegel und 1-Pegel gleich lang ist. Dieser Modus, der einem symmetrischen Taktgenerator gleicht, wird im PC zur Kontrolle der Uhrzeit und zum Erzeugen der Töne für den Lautsprecher verwendet.

Die letzten beiden zur Verfügung stehenden Modi 4 und 5 ähneln in ihrer Funktion den Betriebsarten 0 und 1. Allerdings ist die Auswirkung auf die Zustände der Signalleitungen verschieden, denn in Modus 4 und 5 liegt die OUT-Leitung für die Dauer der vorgegebenen Impulse auf 1 und wechselt dann für eine ganze Periode auf 0, um anschließend wieder auf 1 zu wechseln.

Die Besonderheit der beiden über den Gate-Eingang steuerbaren Modi 1 und 5 ist, daß sich ihr Zustand vor dessen Beendigung auffrischen und danach neustarten (retriggern) läßt. Dabei bleibt in diesem Fall der Ausgang OUT nach der letzten positiven Flanke an dem Gate-Eingang während der gesamten Zähldauer in seinem vorgegebenen Zustand.

Im Modus 1 entspricht dies einem 0-Pegel und in Modus 5 einem 1-Pegel. Dies bedeutet, daß jede neu ankommende Signalflanke den Zählvorgang von neuem startet. In allen Modi, ausgenommen dem Modus 1, können Sie den Zähler so lange arbeiten lassen, wie der Gate-Eingang auf 1 liegt.

Die Programmierung der drei Zählerkanäle des Zeitgebers ist relativ ein-

Das Steuerregister des Zeitgebers	
Zustand	**Funktion**
Bit 0 = 0	16-Bit-Binärzähler ist eingeschaltet
Bit 0 = 1	BCD-Zähler ist eingeschaltet
Bit 1, 2, 3 = 000	Modus 0 ist eingestellt
Bit 1, 2, 3 = 001	Modus 1 ist eingestellt
Bit 1, 2 = 10	Modus 2 ist eingestellt
Bit 1, 2 = 11	Modus 3 ist eingestellt
Bit 1, 2, 3 = 100	Modus 4 ist eingestellt
Bit 1, 2, 3 = 101	Modus 5 ist eingestellt
Bit 4, 5 = 00	Augenblicklichen Zählerstand zum Lesen übernehmen
Bit 4, 5 = 01	Lies oder schreibe das niederwertige Byte (LSB)
Bit 4, 5 = 10	Lies oder schreibe das höherwertige Byte (MSB)
Bit 4, 5 = 11	Zuerst das LSB, dann das MSB lesen oder schreiben
Bit 6, 7 = 00	Zähler 0 ist angewählt
Bit 6, 7 = 01	Zähler 1 ist angewählt
Bit 6, 7 = 10	Zähler 2 ist angewählt
Bit 6, 7 = 11	Konfiguration ist unzulässig

Tabelle. Der Aufbau des Steuerregisters im Zeitgeber 8253

fach, weil die Registerstruktur sehr übersichtlich ist und nicht allzu viele unterschiedliche Funktionen pro Kanal zuläßt. Die Portadressen für die Programmierung des Zeitgebers liegen bei 040 bis 043hex.

Das Register von Kanal 0 besitzt die Portadresse 040hex. Er ist für die Zeit und Einstellung des Datums der Systemuhr zuständig. Das unter der Adresse 041hex ansprechbare Register des Kanal 1 sorgt für das Refresh-Signal zum DMA-Kanal 0. Die Steuerung des Rechtecksignals für den Lautsprecher erfolgt schließlich über den Kanal 2, dessen Register über die Portadresse 042hex angesteuert wird.

Für die Kontrolle der verschiedenen Funktionen des Zeitgebers ist das Steuerregister zuständig, das über die Portadresse 043hex angesprochen wird. Die Tabelle zeigt die Bedeutung der Bits des Steuerregisters im Überblick. Zur Programmierung des Zeitgebers müssen Sie den Baustein zuerst initialisieren, indem Sie für jeden zu steuernden Kanal ein Moduskontrollbyte in das Steuerregister schreiben. Über die Bit 6 und 7 bestimmen Sie den Kanal, für den das jeweilige Moduskontrollbyte vorgesehen ist.

Für den Fall, daß Sie einmal den 16-Bit-Wert aus einem Zählerregister lesen wollen, hat es sich als sinnvoll erwiesen, vor dem Lesen der beiden Bytes zuvor einen Befehl zum Zwischenspeichern zu geben, den Sie mit dem Wert 0 in den Bit 4 und 5 des Steuerregisters erreichen. Unterlassen Sie diese Maßnahme, so kann es unter Umständen der Fall sein, daß der Zählerstand beim Lesen des nieder- und höherwertigen Bytes unterschiedlich ist, weil der Zähler während des Lesevorgangs weiterarbeitet.

Zum Schluß noch ein praktisches Beispiel: Möchten Sie, daß der Lautsprecher Ihres PC mit Hilfe der von Kanal 2 erzeugten Rechteckwellen einen Ton von sich gibt, so müssen Sie den Kanal B des Parallel-Portbausteins 8255 über Portadresse 061hex ansteuern und dessen Bit 0 und 1 mit

```
out &H61,(inp(&H61) or 3)
```

für die gewünschte Dauer des Tons auf 1 setzen.

Um einen bestimmten Ton zu erzeugen, schalten Sie zunächst den Lautsprecher des PC aus, indem Sie die Bit 0 und 1 des Kanal B im Parallel-Portbaustein mit

```
out &H61,(inp(&H61) and 252
```

löschen, ohne jedoch die anderen Bits zu verändern. Für die Tonausgabe muß die Bitfolge 10110110 (B6hex) in das Steuerregister des Zeitgebers geschrieben werden. Dies geschieht mit

```
out &H43,&HB6
```

Nun erwartet der Zeitgeber die Übergabe des 16-Bit-Divisors, mit dem er die Eingangsfrequenz teilt. Diesen Wert berechnen Sie gemäß der extern anliegenden Taktfrequenz von 1,193180 MHz nach der Formel

```
Wert=1193180/Frequenz
```

Das Ergebnis müssen Sie nun nur noch in das höher- und niederwertige Byte zerlegen, weil der 8253 die Daten nur byteweise empfangen kann. Geben Sie zunächst das niederwertige und dann das höherwertige Byte auf die Portadresse des Kanal 2 aus. Beispielsweise den Wert 1000 mit

```
out &H42,&HE8, out &H42,&H03
```

Nun liegt am Ausgang des Zeitgebers die gewünschte Frequenz an. Um sie am Lautsprecher auszugeben, setzen Sie Bit 0 und 1 des Kanals B im Parallel-Portbaustein mit

```
out &H61,(inp(&H61) or 3)
```

wieder auf 1.

(Martin Hepp/ma)

Eine Bitte an unsere Abonnenten

Vermerken Sie bei Schriftverkehr und Zahlungen neben der vollständigen Anschrift stets Ihre Abo-Nummer.

Sie vermeiden damit unnötige Verzögerungen bei der Bearbeitung Ihres Abonnements.

Vielen Dank

Ihre DMV-Versandabteilung

Profi-Tools für QuickBASIC und Turbo BASIC

Schreiben Sie schnellere, leistungsfähigere und professionellere Programme! Wir helfen Ihnen dabei mit nützlichen Tools.

Zum Beispiel:

- **Toolboxen (Fenstertechnik, Menüs, DOS-Funktionen etc.)**
- **Relationale Datenbank mit komfortablem Masken-Editor**
- **Grafik-Paket (Geschäftsgrafik und Zeichensatz-Generator)**
- **Maus-Programmierung**
- **Laserdrucker-Unterstützung**

Alle Pakete mit ausführlich dokumentierten Quelltexten und Programmbeispielen. Wo erforderlich, kommen schnelle Assembler-Routinen zum Einsatz. Wollen Sie mehr aus Ihrem BASIC-Compiler herausholen? Wir informieren Sie gerne kostenlos!

Ingenieur-Büro Harald Zoschke
Berliner Str. 3
D-2306 Schönberg/Holstein
Telefon 0 43 44/61 66

Eingetr. Warenzeichen:
QuickBASIC: Microsoft; Turbo BASIC: Borland

Der Disketten-Controller 765

Bei der Elektronik des Disketten-Controller und Diskettenlaufwerks fehlt oft auch dem fortgeschrittenen PC-Anwender der Durchblick, denn die zum Verständnis notwendigen technischen Informationen sind nur schwer oder gar nicht erhältlich.

Disketten sind nach wie vor das wichtigste Speichermedium für den PC, auch wenn die Festplatte und das CD-ROM dem ehrwürdigen Datenträger massiv Konkurrenz machen. Durch die Einführung der robusten 3 1/2-Zoll-Disketten, die 720 KByte oder gar 1,44 MByte an Daten aufnehmen können, ist die Attraktivität dieses Speichermediums sogar noch aufgewertet worden.

Wenn Sie sich vorstellen, daß auf den handlichen Datenscheiben bis weit über eine Million Zeichen Platz finden, können Sie erahnen, welche aufwendige Mechanik und Elektronik dahintersteckt. Um die Mechanik sollen Sie sich aber keine Gedanken machen, denn hier gibt es nichts zu basteln oder zu programmieren, ganz im Gegenteil: Schon ein kleiner Eingriff in die Mechanik eines Diskettenlaufwerks kann wichtige Justierungen verstellen, so daß das Laufwerk hinterher nicht mehr korrekt arbeitet. Anders dagegen bei der Elektronik. Hier lassen sich die Bausteine nach Herzenslust programmieren und manipulieren. Allerdings sollten Sie für Ihre ersten Versuche nur Disketten mit nicht mehr benötigten Daten verwenden – oder zumindest vorher Sicherheitskopien herstellen.

Das Blockschaltbild in Bild 1 zeigt, wie der Disketten-Controller und das Diskettenlaufwerk zusammenarbeiten. Der Controller-Teil des Blockschaltbildes besteht, am linken Bildrand beginnend, zunächst einmal aus dem Systembus des PC, der Adreß-, Daten- und Steuerleitungen vereint. Diese Leitungen werden über einen Puffer an den Controller-Baustein UPD 765 und das Laufwerk-Auswahlregister geführt.

Das Laufwerk-Auswahlregister spricht über einen zwischengeschalteten Decoder eines von vier Diskettenlaufwerken an, während der Controller-Baustein über vier Steuerleitungen (Ausgänge) und drei Statusleitungen (Eingänge) direkt mit dem Diskettenlaufwerk verbunden ist.

Im (seriellen) Daten-Schreibkanal des Disketten-Controllers ist zwischen Controller-Baustein und Diskettenlaufwerk zusätzlich eine Schreibschaltung eingefügt, und im (seriellen) Daten-Lesekanal befindet sich entsprechend eine Leseschaltung. Beide Schaltungen sind auch mit einem Takt- und Zeitgeber verbunden, der für das richtige Timing bei der seriellen Datenübertragung sorgt.

Beim Schreiben von Daten auf Diskette erzeugt die Schreibschaltung aus den zu schreibenden Daten (»OUT«) und einem Taktsignal WCLK das endgültige Schreibsignal für den Schreib-/Lesekopf des Diskettenlaufwerks. Die Leseschaltung stellt dann beim Einlesen der Signale vom Schreib-/Lesekopf mit Hilfe des Taktsignals RCLK wieder die ursprünglichen Daten (»INP«) her, so daß der Controller-Baustein exakt die Daten erhält, die er zuvor auf die Diskette geschrieben hat.

Auf der Laufwerksseite des Blockschaltbildes befindet sich die Diskette mit Laufwerksmechanik (als Zylinder dargestellt). Für die Pegelanpassung der Signale vom und zum Disketten-Controller sind Schreib- und Leseverstärker für die Schreib-/Lesekopf-Signale und zwei Verstärker für die Decodersignale zum Einschalten des Laufwerksmotors und zum Auswählen des Laufwerks vorhanden. Die Schreib- und Leseverstärker sind besonders wichtig, weil der Disketten-Controller mit TTL-Pegeln (+ 5 Volt und 0 Volt) arbeitet, während die Signale des Schreib-/Lesekopfes schwankende Amplituden haben und geringere Spannungspegel aufweisen.

Verstärker sorgen für die Anpassung der unterschiedlichen Signalpegel von Controller und Laufwerk

Die Programmierung des Disketten-Controllers erfolgt wie bei den anderen Bausteinen des PC auch über die Leitungen des Adreß-, Daten- und Steuerbus. Zuerst muß eine Einheit des Disketten-Controllers über eine Portadresse angesprochen werden, dann zeigen die Signale des Steuerbus an, ob Daten gelesen oder geschrieben werden sollen. Über den Datenbus schließlich werden die Daten übertragen.

Zum Ansprechen über eine Portadresse stehen auf dem Disketten-Controller das Laufwerk-Auswahlregister unter der Portadresse 3F2hex und der Controller-Baustein unter den beiden Portadressen 3F4hex und 3F5hex zur Verfügung.

Das Laufwerk-Auswahlregister besteht aus einem 8-Bit-Register und läßt sich nur beschreiben. Wie der

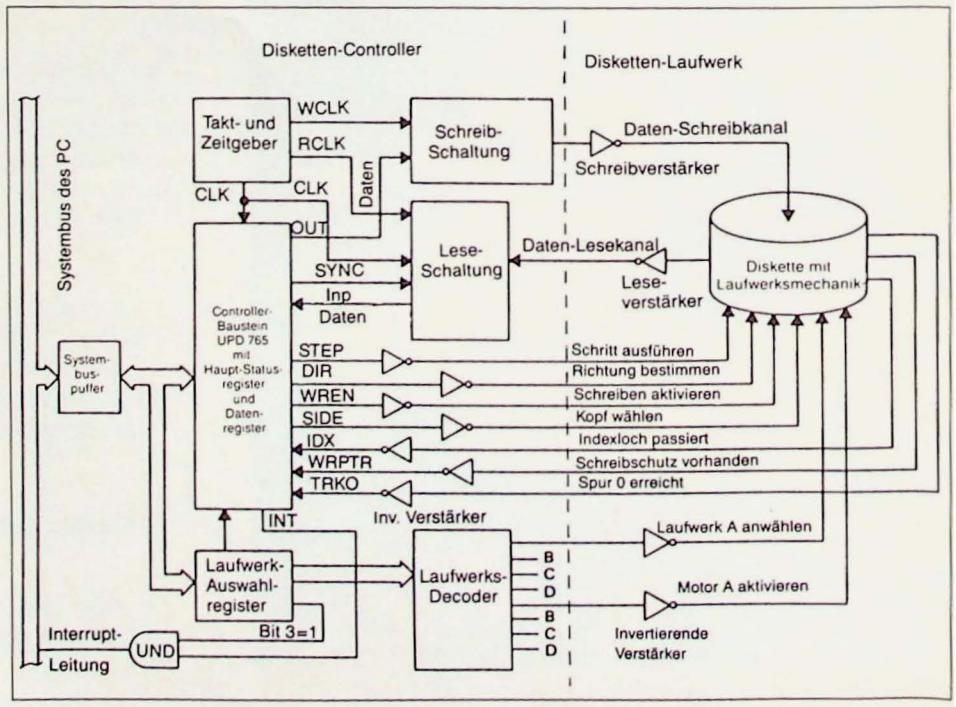


Bild 1. Das Blockschaltbild zeigt den Aufbau eines Disketten-Controllers mit Diskettenlaufwerk

Name schon sagt, wählt das Laufwerk-Auswahlregister eines von maximal vier Laufwerken aus.
Die Bit 0 und 1 definieren, welches Laufwerk ausgewählt ist, und die Bit 4 bis 7 bestimmen, welche Laufwerksmotoren eingeschaltet sind. Die restlichen Bit 2 und 3 sind Steuersignale. Tabelle 1 zeigt die Bedeutung der Bits des Laufwerk-Auswahlregisters im Detail.
Im Controller-Baustein läßt sich über die Portadresse 3F4hex das Haupt-Statusregister und über die Portadresse 3F5hex das 8-Bit-Datenregister ansprechen. Der Inhalt des Haupt-Statusregisters kann nur ausgelesen werden und gibt über aktive Laufwerke und Betriebsmodi Auskunft. Tabelle 2 zeigt im einzelnen die Bedeutung der Bits des Haupt-Statusregisters.
Das Datenregister des Controller-Bausteins besteht eigentlich aus mehreren Registern, die einen Registerstapel bilden und von denen jeweils nur ein Register am Datenbus des PC anliegt. In den Datenregisterstapel lassen sich Daten, Befehle und Parameter schreiben. Das Auslesen dieses Registerstapels ist ebenfalls möglich. Dann liefern die Register diejenigen Daten, die von der Diskette gelesen wurden, oder Statusinformationen über die angeschlossenen Diskettenlaufwerke.

Das Datenregister des Controller-Bausteins besteht in Wirklichkeit aus einem ganzen Registerstapel

Das Blockschaltbild in Bild 1 zeigt deutlich, wie das Laufwerk-Auswahlregister einen Decoder ansteuert, der wiederum das gewünschte Laufwerk (A, B, C oder D) anspricht. Aus Platzgründen ist hier nur das Laufwerk A dargestellt.
Das Interrupt-Signal für die Interrupt-Anforderung des Disketten-Controller wird über eine logische Und-Verknüpfung aus Bit 3 des Laufwerk-Auswahlregisters und dem INT-Ausgang des Controller-Bausteins gebildet. Auf diese Weise läßt sich der Disketten-Controller vom Interrupt-Controller vollständig abkoppeln. Dies ist manchmal notwendig, weil der Interrupt-Controller es nicht erlaubt, eine bestimmte Interrupt-Anforderung völlig zu ignorieren.
Vom Controller-Baustein führen vier Steuersignale direkt zum Diskettenlaufwerk und drei Statusleitungen vom Laufwerk zum Controller zurück. Die Signalpegel der Controller- und Laufwerkssignale werden über invertierende Verstärker (im Bild dargestellt als Dreiecke mit kleinen Kreisen an der Spitze) aneinander angepaßt. Invertierend bedeutet, daß

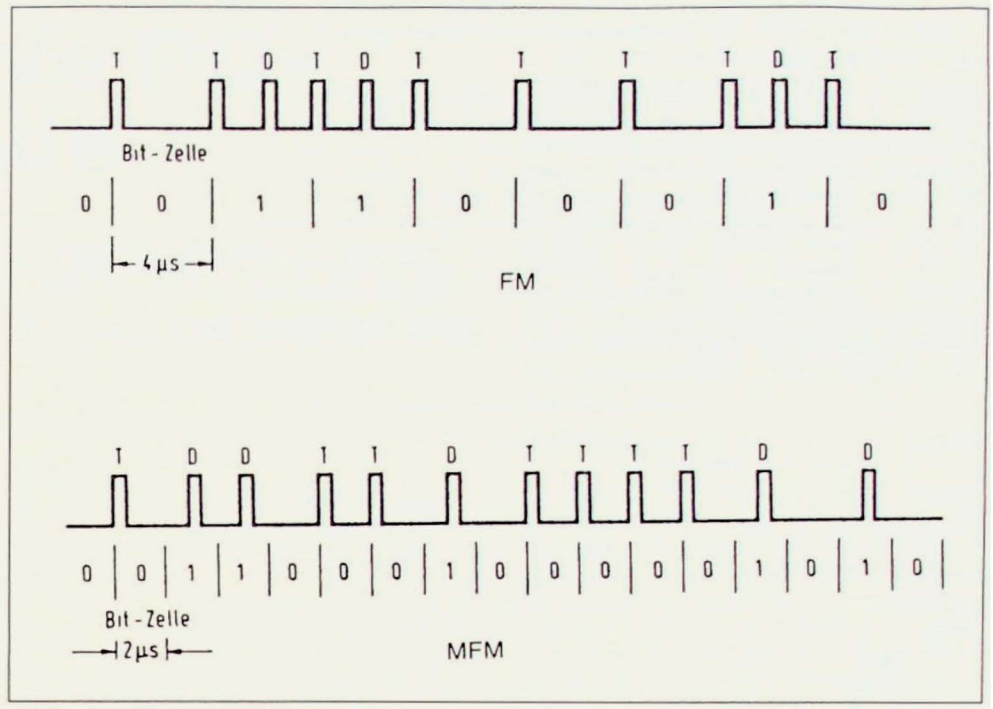


Bild 2. Der Vergleich zwischen dem FM- und MFM-Aufzeichnungsformat zeigt die Überlegenheit des MFM-Verfahrens

die logischen Pegel der Signale vertauscht werden, das heißt, aus logisch 0 wird logisch 1, und aus logisch 1 wird logisch 0. Die Invertierung ist für das korrekte Zusammenspiel von Controller- und Laufwerkssignalen erforderlich, weil die Laufwerkssignale ausschließlich mit negativer Logik arbeiten.
Die vier Signale, die vom Controller zum Laufwerk führen, sind STEP, DIR, WREN und SIDE. Die Bedeutung dieser Signale ist mit einigen Englischkenntnissen bereits den Namen zu entnehmen. Das STEP-Signal bewegt den Schreib-/Lesekopf des Diskettenlaufwerks um einen Schritt (eine Spur) weiter. Jeder Impuls auf dieser Leitung bewirkt einen einzelnen Schritt.
Damit das Diskettenlaufwerk auch erfährt, ob der Schreib-/Lesekopf nach innen oder außen bewegt werden soll, gibt das DIR-Signal (»Direction«) die Richtung der Bewegung an. Ist das DIR-Signal gesetzt, so bewegt sich der Kopf in Richtung Diskettenmitte, und liegt die DIR-Leitung auf 0, so wandert der Schreib-/Lesekopf zum Diskettenrand.
Über das Signal WREN (»Write Enable«) wird das Schreiben von Daten auf Diskette freigegeben. Das SIDE-Signal gibt schließlich die Nummer des Schreib-/Lesekopfes an, der die Daten schreiben oder lesen soll. Liegt die SIDE-Leitung auf 0, so ist der Kopf 0 angesprochen, ist das SIDE-Signal auf 1 gesetzt, so ist entsprechend Schreib-/Lesekopf 1 an der Reihe.
Die drei Leitungen, die vom Laufwerk zum Controller führen, sind die Signale IDX, WRPRT und TRK0. Hier ist die Identifizierung der Signalbedeutung schon etwas schwieriger, weil die Abkürzungen kryptischer geworden sind.
Über das IDX-Signal zeigt das Diskettenlaufwerk an, daß das Indexloch

Die Bits des Laufwerk-Auswahlregisters	
Bit	**Funktion**
Bit 1 = 0 und Bit 0 = 0	Laufwerk A ausgewählt
Bit 1 = 0 und Bit 0 = 1	Laufwerk B ausgewählt
Bit 1 = 1 und Bit 0 = 0	Laufwerk C ausgewählt
Bit 1 = 1 und Bit 0 = 1	Laufwerk D ausgewählt
Bit 2 = 0	Diskettensteuerung zurückgesetzt
Bit 2 = 1	Diskettensteuerung aktiviert
Bit 3 = 0	Software-Interrupt 0Ehex durch Disketten-Controller gesperrt
Bit 3 = 1	Software-Interrupt 0Ehex durch Disketten-Controller freigegeben
Bit 4 = 1	Motor von Laufwerk A arbeitet, Laufwerk kann ausgewählt werden
Bit 5 = 1	Motor von Laufwerk B arbeitet, Laufwerk kann ausgewählt werden
Bit 6 = 1	Motor von Laufwerk C arbeitet, Laufwerk kann ausgewählt werden
Bit 7 = 1	Motor von Laufwerk D arbeitet, Laufwerk kann ausgewählt werden

Tabelle 1. Über das Laufwerk-Auswahlregister wird eines von vier Laufwerken aktiviert

Die Bits des Haupt-Statusregisters	
Bit	**Funktion**
Bit 0 = 1	Der Schreib-/Lesekopf von Laufwerk A wird positioniert
Bit 1 = 1	Der Schreib-/Lesekopf von Laufwerk B wird positioniert
Bit 2 = 1	Der Schreib-/Lesekopf von Laufwerk C wird positioniert
Bit 3 = 1	Der Schreib-/Lesekopf von Laufwerk D wird positioniert
Bit 4 = 1	Ein Schreib- oder Lesebefehl wird verarbeitet
Bit 5 = 1	Die Diskettensteuerung befindet sich nicht im DMA-Modus
Bit 6 = 0	Datenübertragung vom Prozessor zum Datenregister des Disketten-Controllers
Bit 6 = 1	Datenübertragung vom Datenregister des Disketten-Controllers zum Prozessor
Bit 7 = 1	Datenregister zum Senden/Empfangen von Daten bereit

Tabelle 2. Das Haupt-Statusregister zeigt die Betriebsarten des Disketten-Controllers an

Die Signale des Controller-Steckers		
Signal	**A = Ausgang E = Eingang**	**Bedeutung**
Head Load	A	Schreib-/Lesekopf auf Diskette absenken
Drive Select	A	Auswahl eines angeschlossenen Laufwerks
Index	E	Indexloch hat Lichtschranke passiert
Motor On	A	Laufwerksmotor einschalten
Direction	A	Richtungsangabe für Step-Signal
Step	A	Schreib-/Lesekopf eine Spur weiterbewegen
Write Data	A	Daten für Diskette
Write Enable	A	Schreiben auf Diskette freigeben
Track 00	E	Lesekopf hat Spur 0 erreicht
Write Protect	E	Diskette schreibgeschützt
Read Data	E	Daten von Diskette lesen
Side Select	A	Auswahl der Diskettenseite
Ready	E	Laufwerk ist bereit

Tabelle 3. Die 34 Signale des Controller-Steckers arbeiten ausschließlich mit negativer Logik

der Diskette die interne Index-Lichtschranke passiert hat. Durch das Indexloch wird die Lichtschranke nämlich kurz freigegeben und liefert dadurch einen Impuls über die IDX-Leitung an den Disketten-Controller.

Über die Leitung WRPRT (»Write Protected«) teilt das Diskettenlaufwerk dem Controller mit, daß die aktuell im Laufwerk befindliche Diskette schreibgeschützt ist. Jede Diskette besitzt oben rechts eine Kerbe, die sich mit einem Aufkleber verschließen läßt. Dort, wo sich an der Diskette die Kerbe befindet, sitzt im Diskettenlaufwerk eine Schreibschutz-Lichtschranke, hin und wieder auch ein Mikroschalter. Verdeckt nun ein Aufkleber diese Kerbe, so ist die Schreibschutz-Lichtschranke im Laufwerk gesperrt beziehungsweise der Mikroschalter geschlossen, was über das WRPRT-Signal angezeigt wird.

Das dritte und letzte Signal lautet TRK0 (»Track 0«). Über die TRK0-Leitung zeigt das Diskettenlaufwerk dem Controller an, daß sich der Schreib-/Lesekopf auf Spur 0 der Diskette befindet. Hier muß sich das Inhaltsverzeichnis der Diskette und bei Systemdisketten auch der Bootsektor für das Betriebssystem befinden. Die Spur 0 ist deshalb für den Disketten-Controller von besonderer Bedeutung.

Wie Sie dem Blockschaltbild in Bild 1 bereits entnehmen konnten, ist in den Daten-Schreib- und -Lesekanal je eine Schreib- beziehungsweise Leseschaltung eingefügt. Diese beiden Schaltungen sind erforderlich, weil auf einer Diskette nicht die reinen Daten des Controllers aufgezeichnet werden können. Vielmehr ist noch ein spezieller Takt notwendig, den beim Disketten-Controller ein Takt- und Zeitgeber liefert.

Im Blockschaltbild ist der Takt zum Schreiben der Daten durch das Signal WCLK gekennzeichnet und der Takt zum Lesen der Daten durch das Signal RCLK. Zusätzlich benötigt die Leseschaltung noch den Originaltakt CLK, mit dem auch der Controller-Baustein arbeitet. Über die SYNC-Leitung synchronisiert der Controller-Baustein die Datenübertragung mit der Leseschaltung.

Der Aufbau der Schreib- und Leseschaltung hängt vom verwendeten Verfahren für die Aufzeichnung der Daten auf Diskette ab. Die bekanntesten Aufzeichnungsarten sind das FM-Verfahren (Frequenzmodulation) und das MFM-Verfahren (modifizierte Frequenzmodulation). Während das FM-Verfahren nur bei der Datenaufzeichnung auf 8-Zoll-Disketten eine Rolle spielt, hat sich das MFM-Verfahren inzwischen für 5 1/4- und 3 1/2-Zoll-Disketten weltweit durchgesetzt.

Die modifizierte Frequenzmodulation MFM arbeitet gegenüber dem FM-Verfahren mit einer größeren Genauigkeit, so daß sich die Speicherkapazität einer Diskette mit diesem Verfahren verdoppeln läßt. Jede Bit-Information auf der Diskette besitzt beim MFM-Verfahren eine Länge von nur 2 Mikrosekunden. Besitzt ein Bit den Wert 1, so erfolgt die Aufzeichnung eines Impulses in der physikalischen Mitte der Bitzelle. Besitzt das Bit dagegen den Wert 0, dann erfolgt die Aufzeichnung des Impulses am Anfang der Bitzelle. Dies geschieht jedoch nur, wenn das vorherige Bit nicht den Wert 1 hatte. Besteht eine Impulsfolge nur aus 0-Werten, so entspricht dies dem Aufzeichnungs-Taktsignal T.

Bild 2 zeigt den Unterschied in der Aufzeichnungsart und Datendichte zwischen dem alten FM- und dem modernen MFM-Verfahren. Das T gibt jeweils das Taktsignal an.

Um ein Diskettenlaufwerk an den Disketten-Controller eines PC anzuschließen, haben sich die Controller- und Laufwerkshersteller auf einen einheitlichen Standard geeinigt. An jedem Diskettenlaufwerk befindet sich eine 34polige Steckleiste in Form eines Platinensteckers. Der Disketten-Controller wird über ein 34adriges Flachbandkabel an die passende Buchse angeschlossen. Bild 3 zeigt die Belegung des Steckers und Tabelle 3 gibt die Bedeutung aller Anschlüsse aus der Sicht des Controllers an. Sämtliche Signale arbeiten aufgrund der invertierenden Verstärker mit negativer Logik, das heißt, ein Massepegel kennzeichnet den aktiven Zustand dieser Signale.

(Martin Hepp/ma)

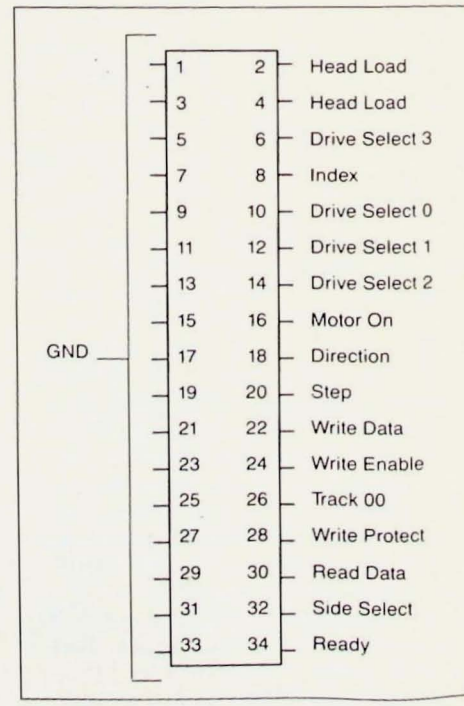


Bild 3. Die Pinbelegung am 34poligen Stecker des Disketten-Controllers

Die parallele PC-Schnittstelle

Die parallele Schnittstelle des PC wird fast ausschließlich für den Anschluß eines Druckers verwendet. Wie diese Schnittstelle funktioniert, wie sie aufgebaut ist und wie sie sich programmieren läßt, erklärt der folgende Beitrag.

Die parallele Schnittstelle im PC ist nach dem vom amerikanischen Druckerhersteller Centronics geschaffenen gleichnamigen Industriestandard aufgebaut, so daß sich beinahe jeder handelsübliche Drucker daran anschließen läßt.
Die Centronics-Schnittstelle arbeitet mit TTL-Pegeln (+ 5 Volt und 0 Volt); daher darf die Länge der Leitung zwischen Computer und Drucker nicht mehr als zwei Meter betragen. Bei Leitungslängen über 0,8 Meter ist eine sorgfältige Abschirmung der Signale untereinander und gegen äußere Einstrahlungen besonders wichtig, um Datenverluste und Übertragungsfehler zu vermeiden.
Die Steckerbelegung der Centronics-Schnittstelle vereinfacht die Fertigung von abgeschirmten Kabeln, denn hier ist jeder Datenleitung und den wichtigsten Steuerleitungen eine eigene Masseleitung zugeteilt. Die Signal- und Masseleitungen sind zudem so angeordnet, daß sie sich am Stecker beziehungsweise an der Buchse direkt gegenüberliegen. Bei Verwendung von Flachbandkabeln liegt so immer eine Masseleitung zwischen zwei Signalleitungen. Der Anschluß des Centronics-Kabels am Drucker erfolgt über einen Amphenol-Stecker (Bild 1).
Dieser Stecker läßt sich mit zwei Haltebügeln an der Centronics-Buchse des Druckers befestigen, so daß sich eine mechanisch stabile Verbindung herstellen läßt, die auch größeren Zugbelastungen widersteht.
Die Tabelle zeigt die Pinbelegung des Amphenol-Steckers. Die Signale mit einem Stern vor der Bezeichnung arbeiten mit negativer Logik, das heißt, sie liegen im aktiven Zustand auf Masse. Das höchstwertige Datenbit 7 der Centronics-Schnittstelle wird von einigen Druckern als Dateninformation ignoriert, weil der einfache ASCII-Zeichensatz ebenfalls nur mit sieben Bit arbeitet.
Allerdings sind solche Drucker für den Anschluß an einen PC weniger geeignet, weil sie nur einen Teil des PC-Zeichensatzes drucken können. Der PC-Zeichensatz ist nämlich gegenüber dem genormten ASCII-Zeichensatz von 128 Zeichen um weitere 128 Sonderzeichen wie Grafiksymbole und landesspezifische Sonderzeichen (zum Beispiel Umlaute) erweitert, und um diese zusätzlichen Zeichen ansprechen zu können, sind alle acht Bit notwendig.
Für die reibungslose Kommunikation zwischen Drucker und PC dient eine Vielzahl von Signalen, deren Funktion im folgenden erläutert wird:

Die Steuerleitungen der Centronics-Schnittstelle sorgen für eine reibungslose Kommunikation

Data Strobe
Wenn der Computer Daten an den Drucker übergeben möchte, aktiviert er für 1 bis 5 Mikrosekunden diese Leitung als Signal für den Drucker, daß Daten zur Übergabe bereitstehen. Data Strobe liegt im aktiven Zustand auf Masse.

Data 0 bis Data 7
Über diese acht Datenleitungen liest der Drucker nach einem Data-Strobe-Signal die anstehenden Daten ein.

Acknowledge
Sobald der Drucker die eingelesenen Daten verarbeitet hat, legt er auf die Acknowledge-Leitung einen Masseimpuls von maximal 10 Mikrosekunden Länge, um dem Computer mitzuteilen, daß er neue Daten übergeben kann.

Busy
Während der Drucker Daten einliest oder ausdruckt, ist das Busy-Signal aktiv und liegt auf + 5 Volt. Auch bei der Initialisierung, im Off-Line-Zustand und beim Auftreten eines Fehlers (siehe Error-Signal) ist das Busy-Signal gesetzt. Bild 2 zeigt das Zusammenspiel von Data Strobe, Acknowledge und Busy bei der Datenübertragung.

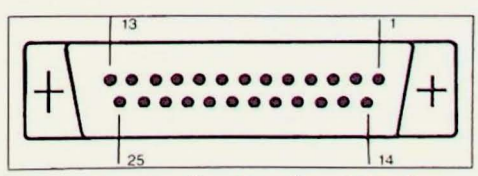


Bild 1. Der 25polige Amphenol-Stecker führt die Signale der Centronics-Schnittstelle an den Drucker

Paper out
Dieses Signal zeigt an, daß das Papierende erreicht ist. Um den Druckvorgang fortzusetzen, muß neues Papier zugeführt werden.

Select
Über dieses Signal zeigt der Drucker dem Computer an, daß er ausgewählt und aktiv ist. Das Signal liegt bei vielen Druckern über einen Pullup-Widerstand konstant auf + 5 Volt.

Autofeed
Über diesen Anschluß läßt sich der Zeilenvorschub steuern. Liegt dieses Signal auf Masse, so führt der Drucker zum Ende jeder Druckzeile einen automatischen Zeilenvorschub aus. Liegt das Autofeed-Signal dagegen auf + 5 Volt, so erfolgt die Steuerung des Zeilenvorschubs über ein Programm beziehungsweise den Druckertreiber. Bei einem PC verbleibt dieses Signal normalerweise auf + 5 Volt.

Error
Dieses Signal liegt auf Masse, sobald der Drucker Off-Line geschaltet, das Papier zu Ende oder ein Fehler während des Druckvorganges aufgetreten ist.

Initial
Über diesen Ausgang kann der Computer den Drucker neu initialisieren, das heißt in den Einschaltzustand zurückversetzen. Dazu muß das Signal für mindestens 50 bis 100 Mikrosekunden auf Masse liegen.

Für eine funktionierende Datenübertragung sind nicht alle Signale des Centronics-Standards erforderlich

Select Input
Über einen Massepegel auf der Select-Input-Leitung erfolgt die Auswahl (Selektierung) des Druckers. Die beiden Auswahlcodes Device Control 1 (DC1) und Device Control 3 (DC3) funktionieren nur, wenn die Select-Leitung inaktiv ist, das heißt auf + 5 Volt liegt. Im anderen Fall läßt sich der Drucker über DC1 und DC3 nicht an- oder abwählen.

Ground
Dies sind die Massepegel zu den Datensignalen und einigen Steuersignalen.

Für einen funktionierenden Druckerbetrieb ist es nicht notwendig, alle Schnittstellensignale zu erzeugen oder auszuwerten. Normalerweise reichen die acht Datenleitungen sowie das Data-Strobe- und das Busy-

Signal aus, um eine ordnungsgemäße Funktion der Centronics-Schnittstelle zu gewährleisten. Die übrigen Anschlüsse bleiben dann unbeschaltet.

Der Ablauf der Datenübertragung vom Computer zum Drucker sieht bei Verwendung eines Minimums an Steuersignalen wie folgt aus:

Zuerst liefert der Computer einen Data-Strobe-Impuls an den Drucker. Sobald dieser Impuls beendet ist, übernimmt der Drucker die auf den Leitungen D0 bis D7 anliegenden Daten. Während er diese Daten einliest, ist das Busy-Signal aktiv. Hat der Drucker das Einlesen der Daten beendet, so ändert auch das Busy-Signal seinen Zustand auf inaktiv. Dies informiert den Computer darüber, daß der Drucker zur Übernahme von neuen Daten bereit ist. Also folgt der zweite Data-Strobe-Impuls, und der Drucker liest das zweite Datenbyte ein.

Während des Einlesens der Daten beginnt der Drucker bereits mit dem Ausdruck. Weil er aber nicht so schnell drucken wie einlesen kann, ist irgendwann der interne Datenpuffer des Druckers gefüllt. Jetzt kann der Drucker immer nur dann ein neues Datenbyte einlesen, wenn er ein Zeichen aus dem Druckpuffer ausgedruckt hat und damit ein Pufferbyte frei geworden ist.

Ein PC arbeitet mit maximal drei parallelen Schnittstellen, die mit »lpt1«, »lpt2« und »lpt3« bezeichnet werden. Für diese drei Schnittstellen sind im Portadressenbereich des PC drei Basisadressen reserviert: 3BChex, 378hex und 278hex.

Beim Selbsttest sucht der PC in dieser Reihenfolge nach parallelen Schnittstellen. Die erste gefundene Schnittstelle wird als »lpt1« verwaltet, die zweite als »lpt2« und die dritte dementsprechend als »lpt3«. Das heißt, der parallelen Schnittstelle, die Sie unter MS-DOS als »lpt1« ansprechen, ist nicht von vornherein eine feste Basisadresse zugewiesen, sondern die erste der drei Basisadressen, die mit einer Schnittstelle belegt ist. Dies ist immer eine der drei obengenannten Adressen, abhängig von der Ausstattung Ihres PC.

So besitzt beispielsweise die parallele Schnittstelle auf der Hercules-Grafikkarte die Basisadresse 3BChex und wird meistens (aus Mangel an weiteren parallelen Schnittstellen) als »lpt1« konfiguriert.

Die Basisadresse einer parallelen Schnittstelle gibt jeweils das Datenregister der Druckerschnittstelle an, in das der 8-Bit-Wert geschrieben wird, der dann an den acht Datenleitungen anliegt. Ein Lesezugriff auf diese Adresse liefert den derzeitigen Zustand der acht Datenleitungen. So lassen sich Fehler in der Schnittstellenkarte leicht ermitteln.

Die auf die Basisadresse unmittelbar folgende Portadresse (3BDhex, 379hex oder 279hex) erlaubt den Zugriff auf das Statusregister. Die Bit 3 bis 7 des Statusregisters entsprechen den Zuständen einiger Signale der Centronics-Schnittstelle:

Bit 3 = 0: Error (Fehler aufgetreten)
Bit 4 = 1: Select (Drucker ausgewählt)
Bit 5 = 1: Paper out (kein Papier vorhanden)
Bit 6 = 0: Acknowledge (fertig)
Bit 7 = 0: Busy (beschäftigt)

Wenn ein Programm Daten an den Drucker sendet, ist es sinnvoll, zuvor und danach dieses Statusregister auszulesen, um zu gewährleisten, daß bei der Übertragung keine Zeichen verlorengehen. Eventuell muß nämlich das Programm den Anwender darauf hinweisen, daß zum Beispiel kein Papier vorhanden ist, der Drucker nicht aktiv ist und ähnliches.

Über die dritte Portadresse der parallelen Schnittstelle (3BEhex, 37Ahex oder 27Ahex) sprechen Sie schließlich das Steuerregister an. Ebenso wie beim Datenregister liefert ein Lesezugriff auf diese Adresse den Status der entsprechenden Signale.

Folgende Funktionen lassen sich durch einen Schreibzugriff auf das Steuerregister beeinflussen:

Bit 0 = 0: Strobe-Signal aktivieren
Bit 1 = 0: Autofeed einschalten
Bit 2 = 0: Drucker initialisieren
Bit 3 = 1: Drucker auswählen
Bit 4 = 1: Acknowledge bewirkt Interrupt 0Fhex (IRQ7)

Die Signale der Centronics-Schnittstelle

Pin	Masse	Computer	Signal
1	19	Ausgang	*Data Strobe (STB)
2	20	Ausgang	Data 0 (D0)
3	21	Ausgang	Data 1 (D1)
4	22	Ausgang	Data 2 (D2)
5	23	Ausgang	Data 3 (D3)
6	24	Ausgang	Data 4 (D4)
7	25	Ausgang	Data 5 (D5)
8	26	Ausgang	Data 6 (D6)
9	27	Ausgang	Data 7 (D7)
10	28	Eingang	*Acknowledge (ACK)
11	29	Eingang	Busy
12	30	Eingang	Paper out
13	–	Eingang	Select
14	–	Ausgang	*Autofeed
15	–	Eingang	*Error
16	–	Ausgang	*Initial
17	–	Ausgang	*Select Input
18 bis 25	–	–	Ground (GND)

Tabelle. Die Belegung der 25poligen Centronics-Schnittstelle

Mit diesen Informationen dürfte es Ihnen nun leichtfallen, die parallele Schnittstelle des PC selbst zu programmieren, um sie beispielsweise zur Ansteuerung von selbstgebauten Schaltungen und anderen Hardware-Erweiterungen zu verwenden.

Wenn Sie allerdings nur Zeichen an den Drucker schicken möchten, dann ist die Verwendung des Drucker-Interrupt 17hex doch einfacher.

(Martin Hepp/ma)

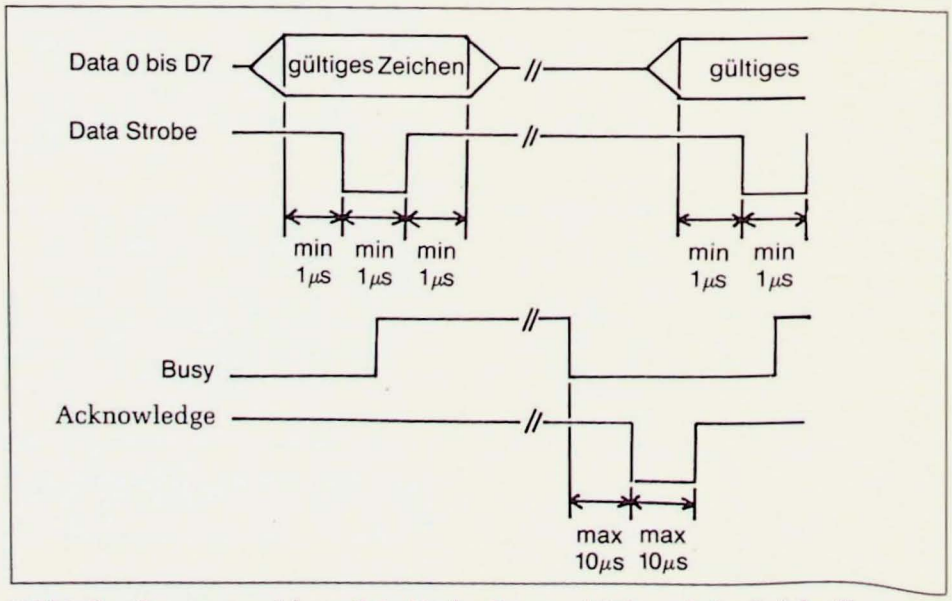
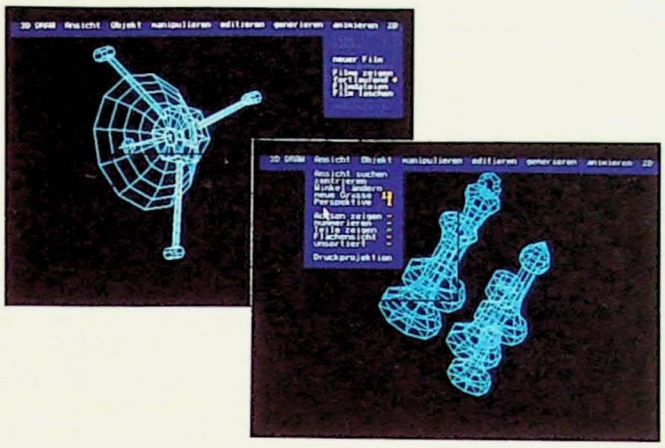


Bild 2. Das Zusammenspiel von Data Strobe, Busy und Acknowledge bei der Datenübertragung vom Computer an den Drucker

DOS SOFTWARE TOOLS

Jetzt neu aus der Software-Schmiede der DOS International
High-Tech-Software zu Low-Cost-Preisen

DOS-Tools Vol. 4 – 3D-Draw



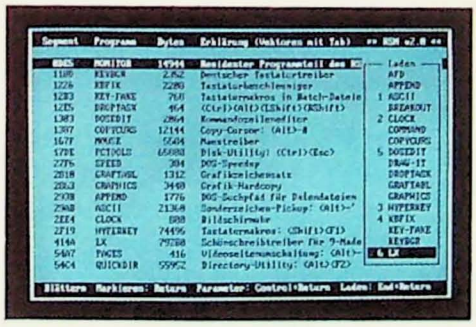
Konstruieren Sie in drei Dimensionen!
Das komfortable CAD-Programm »3D-Draw« bietet verdeckte Flächen und, als Besonderheit, Lichtquellensimulation. Mit der integrierten Animationsroutine erzeugen Sie beliebig lange Filme.

5 1/4-Zoll-Diskette *Best.-Nr. 6007*
3 1/2-Zoll-Diskette *Best.-Nr. 6008*

Nur 99,– DM*

Die Leistungsmerkmale:
* Ansicht durch Eingabe der Projektionswinkel frei wählbar
* Auf Wunsch lassen sich drei verschiedene 2D-Ansichten erzeugen, die am unteren Bildschirmrand eingeblendet werden
* 2D-Editor zur schnellen Konstruktion von Objekten
* Integrierter Taschenrechner zur schnellen Berechnung von Konstruktionsdaten
* Größe und perspektivische Verzerrung lassen sich einstellen
* Kombination von getrennt konstruierten Teilen
* 3D-Funktionen als Manipulationswerkzeug
* Punkte, Kanten und Flächen sind frei editierbar
* Drei Raumachsen mit 1600-Ebenen-Editor
* Verdeckte Flächen
* Lichtquellensimulation
* Animationsroutine mit Größen- und Perspektivenveränderung von beliebiger Länge; ein eigener Animationslinker ist eingeschlossen
* Verformen der Objekte durch Zufallszahl oder Verknüpfung mit mathematischen Formeln
* Erzeugung von Rotations- oder Translationskörpern und modifizierten Duplikaten
* Programmierer-Schnittstelle, Objekte lassen sich mit einem Textprogramm oder Editor generieren
* Vielzahl an Ausgabeformaten: 3D-Draw, Autocad-Script, Postscript, EPS-Format, HPGL-Plot-Format und ASCII-Textdateien
* Alle gängigen Grafikkarten werden unterstützt, komplette Ausnutzung der EGA-Palette mit 64 Farben
* Benutzerführung mit Pulldown-Menüs
* Viele Beispieldateien

Hardwarevoraussetzungen
Computer:
– PC, XT, AT oder Kompatibler mit mindestens 640 KByte Arbeitsspeicher
– ein oder zwei Diskettenlaufwerke beziehungsweise ein Diskettenlaufwerk und eine Festplatte
Grafikkarten:
– Hercules-, CGA-, EGA- oder VGA-Karte
Ausgabemedien:
– Epson-kompatibler Drucker, Ausgabe von Postscript-Dateien in Verbindung mit Postscript-Emulator, Postscript-fähiger Laserdrucker, Plotter mit HPGL-Standard
Sonstiges:
– Microsoft-kompatible Maus unbedingt erforderlich
– MS-DOS ab Version 3.0



DOS-Tools Vol.1 – RSM - Resident-Software-Manager

Programmverwaltung à la carte
Menügesteuertes Löschen und Nachladen von residenten TSR-Programmen löst Ihre Speicherplatzprobleme.
Die Leistungsmerkmale von RSM:
* RSM zeigt den noch freien Arbeitsspeicher
* RSM registriert geladene residente Programme
* RSM erinnert an die Aktivierungstastenkombination
* RSM informiert über benutzte Interrupt-Vektoren
* RSM löscht beliebige speicherresidente Programme
* RSM lädt mehrere TSR-Programme in einem Arbeitsgang
* RSM bietet umfassende, komfortable Menüsteuerung
* RSM besitzt einen universellen Maustreiber
* RSM belegt lediglich 15 KByte Speicherplatz
* RSM informiert mit einem umfangreichen Handbuch

Für alle PCs, XTs, ATs und Kompatiblen mit mindestens 256 KByte Arbeitsspeicher

5 1/4-Zoll-Diskette *Best.-Nr. 6001*
3 1/2-Zoll-Diskette *Best.-Nr. 6002*

Nur 69,– DM*

DOS-Tools Vol.2 – PC-Utilities

Zeitlupe PC
Keine Angst vor Geschwindigkeit! Mit Zeitlupe PC verlangsamen Sie Ihren Computer bis zu einem Bruchteil der Taktfrequenz eines 4,77-MHz-PC. Mit Boot-Option!

Softreset
Spannen Sie Ihre Nerven nicht weiterhin auf die Folter! Mit Softreset bleibt der komplette Speicherinhalt auch nach einem Reset erhalten und kann auf Diskette oder Festplatte gespeichert werden.

Extras
Der universell einsetzbare Tastaturtreiber »T6« mit Quellcode. Passen Sie Ihre Tastatur an Ihre persönlichen Bedürfnisse an!

5 1/4-Zoll-Diskette *Best.-Nr. 6003*
3 1/2-Zoll-Diskette *Best.-Nr. 6004*

Nur 69,– DM*

DOS-Tools Vol.3 – MS-DOS-Utilities

Fünf neue MS-DOS-Befehle, auf die Sie schon lange gewartet haben. Erleichtern Sie sich künftig den Umgang mit MS-DOS. Die Befehle im einzelnen:

Call – verschachtelt Aufrufe von Batch-Dateien vor MS-DOS-Version 3.3.

Code – verschlüsselt Dateien mit einem Paßwort.

Makeroom – löscht Dateien gemäß Suffix in allen Unterverzeichnissen.

Setdir, Restdir – aktivieren nach Programmaufruf das Orginal-Verzeichnis.

Ty – gibt eine Datei als Hex-Dump, im Binärformat oder mit Zeilennummern auf dem Drucker oder dem Bildschirm aus.

Extras – »SCopy«, das professionelle Kopierprogramm, »Filed«, der Editor, dem nichts verborgen bleibt, und das Übersetzungsprogramm »Xlate«.

5 1/4-Zoll-Diskette *Best.Nr. 6005*
3 1/2-Zoll-Diskette *Best.-Nr. 6006*

Nur 49,– DM*

Sämtliche Programme sind wahlweise auf 5 1/4-Zoll- oder 3 1/2-Zoll-Diskette erhältlich. (Bitte bei Bestellung unbedingt angeben.)
Unabhängig von der Anzahl der bestellten Artikel berechnen wir für das Inland 4,– DM bzw. für das Ausland 6,– DM Porto und Verpackung.
*Unverbindliche Preisempfehlung. Bitte benutzen Sie die Bestellkarte.

DMV-Verlag · Abteilung DOS-Software · Postfach 250 · 3440 Eschwege

DMV Daten- und Medienverlag

Die serielle PC-Schnittstelle

Die serielle Schnittstelle des PC eröffnet dem Anwender ein weites Betätigungsfeld. Über diese Schnittstelle lassen sich sowohl Drucker und Plotter als auch Akustikkoppler oder Modems ansteuern. Wegen seiner vielfältigen Fähigkeiten meiden allerdings viele PC-Besitzer die Programmierung der seriellen Schnittstelle.

Für den Anwender ist die serielle Schnittstelle gewissermaßen das Tor zur Außenwelt, denn über sie lassen sich nicht nur die verschiedensten Peripheriegeräte, sondern auch ein Akustikkoppler oder Modem anschließen. Auf diese Weise kann der PC-Besitzer Daten und Programme über die Telefonleitung mit weit entfernten Rechnersystemen austauschen.

In einem PC lassen sich bis zu zwei serielle Schnittstellen einbauen, deren Basisadressen bei 3F8hex (»com1«) und 2F8hex (»com2«) liegen. Mit spezieller Software sind aber auch mehr als zwei serielle Schnittstellen realisierbar.

Im PC besteht die serielle Schnittstelle im wesentlichen aus dem UART-Baustein 8250, der ausnahmsweise nicht vom IBM-Hoflieferanten Intel, sondern von National Semiconductor hergestellt wird. Im AT findet meistens die schnellere Version NS16450 Verwendung, die ansonsten jedoch funktionell dem Baustein 8250 entspricht.

Die Abkürzung »UART« steht für »Universal Asynchronous Receiver/Transmitter«, was in etwa »universeller asynchroner Empfänger/Sender« bedeutet. Oft bezeichnet man einen solchen Baustein auch als »SIO«, das heißt »Serial Input/Output«.

Die Aufgabe des UART-Bausteins besteht darin, die zu sendenden Daten in einzelne Bits zu zerlegen, nacheinander zu übertragen und die eingehenden Bitfolgen wieder zu Bytes zusammenzusetzen.

Bei der seriellen Übertragung werden die Bits hintereinander, also »im Gänsemarsch«, übertragen. Dadurch dauert ein solcher Transfer natürlich acht- bis zehnmal so lange wie bei einer parallelen Übertragung, bei der alle acht Bit und Steuersignale zur gleichen Zeit übergeben werden. Der Vorteil der seriellen Übertragung liegt darin, daß weit weniger Leitungen als bei einer parallelen Übertragung nötig sind. Schon drei Leitungen reichen für die serielle Kommunikation aus: eine Leitung für die Sendedaten, eine für die Emfangsdaten und die Masseleitung als Bezugspotential.

Die serielle Schnittstelle im PC arbeitet nach der internationalen Norm RS 232C, die in Europa auch als V.24 bekannt ist. Diese Definition besagt, daß eine logische 1 durch eine Spannung zwischen -3 und -15 (meist -12) Volt und eine logische 0 durch eine Spannung zwischen + 3 und + 15 (meist + 12) Volt repräsentiert wird. Spannungen von -3 bis + 3 Volt sind keinem der beiden Zustände zugeordnet, diesen undefinierten Bereich bezeichnet man auch als Hysterese.

Die serielle Schnittstelle im PC arbeitet nach der internationalen Norm RS 232C, auch als V.24 bekannt

Bei der seriellen Übetragung werden neben den Leitungen TD (»Transmit Data«, deutsch: Sendedaten), RD (»Receive Data«, deutsch: Empfangsdaten) und GND (»Ground«, deutsch: Masse) noch sechs weitere Signale verwendet. Es handelt sich dabei um die Signale DCD, DTR, DSR, RTS, CTS sowie RI.

Diese Leitungen nennt man Handshake-Leitungen, und ihre Aufgabe ist es, eine störungsfreie Kommunikation zwischen DTE (»Data Terminal Equipment«, deutsch: Datenendgerät, im folgenden mit Computer bezeichnet) und DCE (»Data Communication Equipment«, deutsch: Datenübertragungsgerät, im folgenden mit Modem bezeichnet) zu gewährleisten.

Über diese Leitungen teilt der Computer dem Modem beispielsweise mit, daß er senden möchte, und das Modem informiert den PC, wann es zur Übertragung bereit ist. Oft läßt man diese Leitungen aber auch unbenutzt, weil bei niedrigen Übertragungsgeschwindigkeiten ein Handshaking entbehrlich ist.

Die Tabelle zeigt die nach der EIA-Norm (»Electronic Industries Association«) festgelegten Signale der RS-232C-Schnittstelle. Von dieser Vielzahl an Signalen werden jedoch bei PCs gewöhnlich nur die bereits genannten neun Leitungen benutzt, so daß die RS-232C-Schnittstelle oft auch als 9polige Schnittstelle ausgeführt ist.

Die vollausgebaute RS-232C-Schnittstelle arbeitet dagegen mit 25poligen DSUB-Miniaturbuchsen und -steckern. An der 25poligen Schnittstelle eines PC liegen jedoch in der Regel ebenfalls nur die neun wichtigsten Signale an.

Die RS-232C-Schnittstelle in PCs unterstützt Übertragungsgeschwindigkeiten von bis zu 9600 Baud (Bit pro Sekunde). Dabei dürfen die Übertragungsleitungen maximal 15 Meter lang sein. Übliche Übertragungsraten sind 50, 75, 110, 150, 300, 600, 1200, 2400, 4800 und 9600 Baud. Die neuen Modelle der PS/2-Serie von IBM erlauben dagegen auch Baudraten bis zu 19200 Baud.

Durch direkte Registerprogrammierung lassen sich aber auch bei gewöhnlichen PCs Übertragungsraten von 19200, 38400, 57600 und sogar 115200 Baud erreichen. Allerdings ist dann eine fehlerfreie Übertragung nur bei der Verwendung von sehr kurzen, abgeschirmten Kabeln garantiert. Auch müssen Sie in diesem Fall sämtliche Routinen für die serielle Schnittstelle selbst programmieren, weil das BIOS nur Übertragungsgeschwindigkeiten bis 9600 Baud unterstützt. Verschiedene kommerzielle Programme für den Datenaustausch zwischen Laptop und stationären PCs (zum Beispiel Laplink) machen von dieser Möglichkeit Gebrauch.

Die RS-232- Schnittstelle des PC arbeitet asynchron – die Daten werden ohne ein Taktsignal übertragen

Bei der Verwendung von Modems sind die gebräuchlichsten Übertragungsraten 1200 und 2400 Baud, Akustikkoppler arbeiten meistens mit 300 Baud. Inzwischen gibt es jedoch auch Akustikkoppler, die mit bis zu 2400 Baud arbeiten, und bei Modems sind seit kurzem 9600 Baud die neue Höchstgeschwindigkeit.

Die serielle Schnittstelle eines PC arbeitet asynchron, das heißt, die seriellen Daten werden ohne ein Taktsignal übertragen. Im Ruhezustand führt die Sendeleitung einen -12-Volt-Pegel (logisch 1). Den Beginn der Übertragung signalisiert ein + 12-Volt-Impuls, das sogenannte Startbit. Ein Startbit hat die Länge eines Bit.

Darauf folgen die Datenbits, wobei zwischen fünf und acht Datenbit verwendet werden. Abschließend sendet

der UART-Baustein ein, eineinhalb oder zwei Stoppbit (logisch 1), die das Ende des Bitstroms anzeigen. Wahlweise fügt der UART-Baustein zwischen dem letzten Datenbit und dem ersten Stoppbit noch ein zusätzliches Prüfbit ein. Dieses Bit heißt auch Paritätsbit und dient dazu, eventuelle Übertragungsfehler aufzuspüren.
Es gibt zwei Arten der Paritätsüberprüfung, die gerade und die ungerade Parität. Bei gerader Parität muß die Summe der gesetzten Datenbits und des Paritätsbit gerade, bei ungerader Parität entsprechend ungerade sein. Auf diese Weise kann der Empfänger feststellen, ob das entsprechende Zeichen korrekt übertragen wurde.

Ein gemeinsames Übertragungsprotokoll ist Voraussetzung für die problemlose Datenübertragung

Vor Beginn einer Datenübertragung müssen sich Sender und Empfänger auf die zu verwendenden Parameter einigen. Beide müssen nicht nur die gleiche Baudrate verwenden, sondern auch in der Anzahl der Stoppbits und der Art der Parität (gerade, ungerade oder keine) übereinstimmen.
Dieses Übertragungsverfahren nennt man asynchron, weil Sender und Empfänger sich vor jedem Byte selbsttätig neu synchronsieren. Deshalb wirken sich auch geringfügige Schwankungen der Taktrate nicht störend aus.
Glücklicherweise übernimmt der UART-Baustein der seriellen Schnittstelle fast völlig selbständig die Umwandlung der seriellen in parallele Daten und umgekehrt. Der Programmierer muß nur zu Beginn der Übertragung den Baustein korrekt initialisieren, die zu sendenden Daten nacheinander bereitstellen und alle empfangenen Zeichen rechtzeitig (bevor das nächste Zeichen eintrifft) auslesen.
Zu diesem Zweck verfügt der UART-Baustein über zehn Register, die über Portadressen angesprochen werden. Vier Register des UART-Bausteins legen die Parameter der Übertragung fest. Das Modem-Kontroll-Register bestimmt, welche Handshake-Leitungen Verwendung finden, das Leitungs-Kontrollregister definiert die Anzahl der Daten- und Stoppbits sowie die Art der Parität, und die beiden Baudraten-Register enthalten den Divisor, aus dem die Baudrate gewonnen wird.
Die beiden Baudratenregister bilden zusammen einen 16-Bit-Divisor, der intern mit 16 multipliziert und durch den dann die extern anliegende Taktfrequenz von 1,8432 MHz geteilt wird. Die Baudrate errechnet sich damit folgendermaßen:

```
Baudrate=115200/Wert
```

Bei 1200 Baud beträgt der Wert für das Baudratenregister beispielsweise 96.
Ein Programm kann die serielle Schnittstelle auf zwei unterschiedliche Arten ansprechen: Entweder fragt es laufend den Status des UART-Bausteins ab und übergibt ihm bei Bedarf die zu sendenen Daten beziehungsweise liest die empfangenen Daten aus, oder der UART-Baustein löst immer dann einen Interrupt aus, wenn er die Aufmerksamkeit des Prozessors benötigt. Zu diesem Zweck sind zwei Leitungen des Interrupt-Controller für die beiden seriellen Schnittstellen »com1« und »com2« reserviert.
Die meisten Programme verwenden zur Bedienung der seriellen Schnittstelle das interruptgesteuerte Verfahren, weil dem Prozessor dadurch mehr Zeit für die Bearbeitung anderer Aufgaben bleibt.

Unter welchen Umständen ein Interrupt ausgelöst werden soll, muß das Programm zu Beginn im Interrupt-Erkennungsregister des UART-Bausteins festlegen. Die zu sendenden Daten schreibt das Programm in das Sendedaten-Halteregister, und die empfangenen Daten liest es aus dem Empfangsdatenregister aus. Beide Register können nur ein einzelnes Byte aufnehmen. Deshalb muß das Emfangsdatenregister sofort ausgelesen werden, wenn ein Zeichen eintrifft. Das nächste empfangene Zeichen überschreibt nämlich das vorherige ohne Warnung.
Wie Sie sehen, ist die direkte Programmierung des UART-Bausteins nicht einfach. Erheblich einfacher geht dies in Hochsprachen vonstatten, weil diese Ihnen vorgefertigte Routinen zur Verfügung stellen.
Weitergehende, detaillierte Informationen zu der Programmierung des UART-Bausteins 8250 finden Sie im entsprechenden Datenblatt von National Semiconductor.

(Martin Hepp/ma)

Die Pinbelegung der RS-232C-Schnittstelle

Pin	Kurzbezeichnung	Englische Bezeichnung	Datenendgerät	Bedeutung
1	CHASSIS GND	Protective Ground	–	Schutzerde
2	TxD	Transmit Data	Ausgang	Sendedaten eines Datenendgerätes
3	RxD	Receive Data	Eingang	Empfangsdaten für ein Datenendgerät
4	RTS	Request To Send	Ausgang	Anforderung des Datenendgerätes zum Senden
5	CTS	Clear To Send	Eingang	Datenübertragungsgerät frei zum Übertragen an Gegenstation
6	DSR	Data Set Ready	Eingang	Datenübertragungsgerät betriebsbereit
7	GND	Signal Ground	–	Signalerde
8	DCD	Data Carrier Detect	Eingang	Empfangssignalpegel von Gegenstation für Datenübertragungsgerät
9	–	+ Test Voltage	–	Positive Prüfspannung
10	–	– Test Voltage	–	Negative Prüfspannung
11	–	Select Transmit Frequenzy	Ausgang	Sendefrequenzauswahl
12	2. DCD	Secondary Carrier Detect	Eingang	DCD für Rückkanal
13	2. CTS	Secondary Clear To Send	Eingang	CTS für Rückkanal
14	2. TxD	Secondary Transmitted Data	Ausgang	TxD für Rückkanal
15	TxC	Transmitter Signal Clock	Eingang	Sendetakt von DCE
16	2. RxD	Secondary Received Data	Eingang	RxD für Rückkanal
17	RxC	Receiver Signal Clock	Eingang	Empfangstakt von DCE
18	–	–	–	
19	2. RTS	Secondary Request To Send	Ausgang	RTS für Rückkanal
20	DTR	Data Terminal Ready	Ausgang	Datenendgerät betriebsbereit
21	–	Signal Quality Detect	Eingang	Empfangsqualität von Gegenstation für Datenübertragungseinheit
22	RI	Ring Indicator	Eingang	Klingelanzeige: Dieses Signal ist aktiv, wenn ein Anruf hereinkommt, also wenn das Telefon läutet
23	DRS	Data Rate Select	Ausgang /Eingang	Auswahl der Übertragungsrate für Datenübertragungsgerät/Datenendgerät
24	TxC	Transmitter Signal Clock	Sendetakt von DTE	
25	BUSY	Line Busy	Ausgang	Datenendgerät beschäftigt Datenendgerät: DTE (Data Terminal Equipment) Datenübertragungsgerät: DCE (Data Communication Equipment)

Tabelle. Diese Signale sind für die RS-232C-Schnittstelle durch die EIA-Norm festgelegt

Der Video-Controller 6845

Der Monitor ist neben der Tastatur die wichtigste Schnittstelle zwischen Mensch und Computer. Für den Anschluß eines Monitors enthält deshalb jeder PC eine Grafikkarte, deren Herz ein Video-Controller bildet. Der Monochromadapter, die Hercules- und die CGA-Karte arbeiten mit dem Typ 6845 von Motorola.

Je nach Anwendungsgebiet lassen sich verschiedene Grafikkarten in einem PC installieren. Doch unabhängig davon, ob Sie einen Monochromadapter, eine Hercules-, CGA-, EGA- oder VGA-Karte verwenden – alle diese Karten sind mit einem Video-Controller ausgestattet. Der Monochromadapter sowie die Hercules- und CGA-Karte arbeiten mit dem Video-Controller 6845 von Motorola, während die EGA- und VGA-Karten wegen der komplexeren Aufgaben spezielle hochintegrierte Bausteine verwenden, die allerdings die wichtigsten Funktionen des 6845 emulieren.

Der Video-Controller wird oft auch als CRTC (die Abkürzung für »Cathode Ray Tube Controller«) bezeichnet. Die Aufgabe des Video-Controller besteht darin, die Daten aus dem Bildschirmspeicher in elektrische Signale für den Monitor umzuwandeln.

Diesen Vorgang erläutert ein kurzer Überblick über die Arbeitsweise einer Grafikkarte: Daten, die im PC für den Monitor bestimmt sind, schreibt das entsprechende Programm in den Bildschirmspeicher auf der jeweiligen Grafikkarte.

Je nach Art der verwendeten Grafikkarte befindet sich der Bildschirmspeicher in einem bestimmten Speicherbereich innerhalb des Adreßraums des Prozessors. Tabelle 1 zeigt die Speicherbereiche der wichtigsten Grafikkarten im Vergleich.

Um die Daten im Bildschirmspeicher der Grafikkarte in Punkte, Zeichen oder Grafiken auf dem Bildschirm umzuwandeln, liest der Video-Controller diese Daten aus und formt daraus die elektrischen Signale für den Monitor, der dann den gewünschten Text oder die Grafik anzeigt.

Wenn sich die Grafikkarte im Textmodus befindet, handelt es sich bei den Daten im Bildschirmspeicher um ASCII-Zeichen und Attributbytes. In diesem Fall entnimmt der Video-Controller die Bildpunkte für die ASCII-Zeichen aus einem Zeichengenerator (ein ROM oder EPROM), der sich ebenfalls auf der Grafikkarte befindet. Dieser Zeichengenerator enthält die Bilddaten für alle 256 Zeichen des PC-Zeichensatzes.

Die meisten PC-Grafikkarten arbeiten in den Textmodi mit mehreren Bildschirmseiten

Im Textmodus benötigt ein Bildschirm wesentlich weniger Speicherplatz als im Grafikmodus, weil ein einzelnes Byte sehr viele Bildpunkte auf dem Bildschirm definiert. Um den dadurch zusätzlich vorhandenen Speicherplatz verfügbar zu machen, arbeiten die entsprechenden Grafikkarten in den Textmodi mit sogenannten Bildschirmseiten, das heißt, im Bildschirmspeicher können sich mehrere komplette Bilder gleichzeitig befinden. So arbeitet beispielsweise die CGA-Farbgrafikkarte im Textmodus mit bis zu acht Bildschirmseiten, zwischen denen sich blitzschnell umschalten läßt.

Im Grafikmodus sind die Daten für jeden einzelnen Bildpunkt im Bildschirmspeicher abgelegt, daher ist eine Umwandlung nicht mehr notwendig. Bei Farbgrafikkarten muß der Video-Controller aus den Bilddaten allerdings noch die Farbinformationen auswerten. Der Monochromadapter ist dagegen nicht grafikfähig und erlaubt daher auch keine verschiedenen Bildschirmseiten im Textmodus, und weil die Hercules-Grafikkarte vom BIOS des PC nur so weit wie der Monochromadapter unterstützt wird, gilt für sie das gleiche.

Der Video-Controller einer PC-Grafikkarte arbeitet mit einer Buchstabenmatrix von 8x8 Bildpunkten (CGA-Farbgrafikkarte), 14x8 Bildpunkten (EGA-Karte), 14x9 Bildpunkten (Monochromadapter und Hercules-Grafikkarte) oder sogar 16x9 Bildpunkten (VGA-Karte). Dadurch treten bei Unterlängen keine Probleme auf, und die Zeichen gewinnen mit zunehmender Bildpunktauflösung der Zeichenmatix an Qualität.

Bild 1 zeigt die Arbeitsweise des Video-Controllers im PC. Der Video-Controller kommuniziert über den Daten- und Adreßbus mit dem Prozessor und dem Bildschirmspeicher. Moderne EGA- und VGA-Karten besitzen oft einen Dual-Ported-Bildschirmspeicher, das heißt, Prozessor und Video-Controller können gleichzeitig, jeder über ein eigenes Port, auf den Bildschirmspeicher zugreifen. Dadurch läßt sich der lästige »Schneefall« vermeiden, der ansonsten auftritt, wenn der Prozessor Daten in den Bildschirmspeicher schreiben will, während der Video-Controller gerade Daten auslesen möchte.

Der Video-Controller liest periodisch (je nach Grafikkarte 50-, 60- oder 70mal pro Sekunde) den Inhalt einer Bildschirmseite (Textmodus) oder den gesamten Bildschirmspeicher (Monochromadapter oder Grafikmodus) aus und formt ihn in Signale für

Grafikkarte	Speicherbereich
Monochromadapter	B0000 bis B0FFFhex
Hercules-Grafikkarte	B0000 bis BFFFFhex
CGA-Farbgrafikkarte	B8000 bis BBFFFhex
EGA-Karte	A0000 bis BFFFFhex
VGA-Karte	A0000 bis BFFFFhex

Tabelle 1. Die Bildschirmspeicher der verschiedenen Grafikkarten belegen unterschiedliche Speicherbereiche

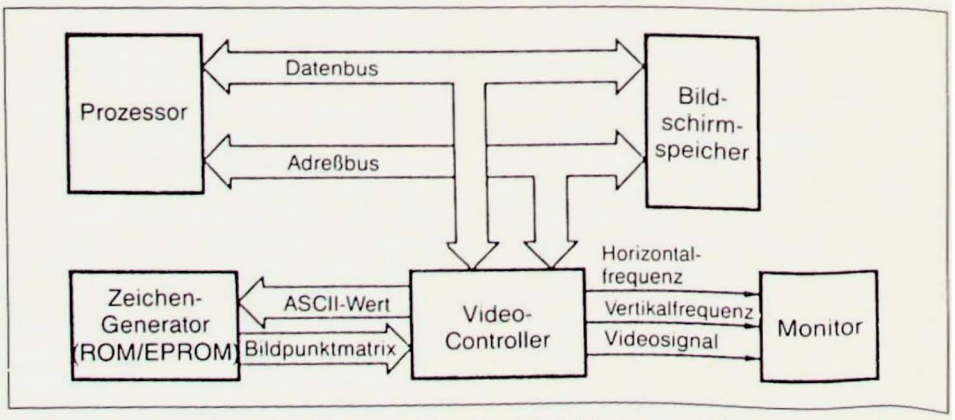


Bild 1. Der Video-Controller kommuniziert im PC mit Prozessor, Bildschirmspeicher, Zeichengenerator und Monitor

Der Registersatz des Video-Controllers 6845

Register	Bit	Voreinstellung				Bedeutung
		MDA	CGA-Textmodus mit 80 Zeichen/Zeile	CGA-Textmodus mit 40 Zeichen/Zeile	CGA-Grafikmodus	
0	0 bis 7	97	113	56	56	Zahl der (teilweise unsichtbaren) Textzeichen pro Textzeile -1 (für HSYNC-Signal)
1	0 bis 7	80	80	40	40	Zahl der tatsächlich angezeigten Textzeichen pro Textzeile
2	0 bis 7	82	90	45	45	Rasterzeilen-Position des HSYNC-Signals (bestimmt horizontale Verschiebung)
3	0 bis 3	15	10	10	10	Länge des HSYNC-Signals
	4 bis 7	0	0	0	0	Länge des VSYNC-Signals (0 = 16 Zeiteinheiten)
4	0 bis 6	25	31	31	127	Zahl der (teilweise unsichtbaren) Textzeilen -1
5	0 bis 4	6	6	6	6	Feineinstellung für Register 4 in Rasterzeilen
6	0 bis 6	25	25	25	100	Zahl der tatsächlich angezeigten Textzeilen
7	0 bis 6	25	28	28	112	Zeichenposition des VSYNC-Signals -1 (bestimmt senkrechte Verschiebung)
8	0 und 1	2	2	2	2	Zeilensprungmodus (Interlace) ein (1 und 3) oder aus (0 und 2)
9	0 bis 4	13	7	7	1	Höhe eines Zeichens in Rasterzeilen -1
10	0 bis 3	11	6	6	6	Erste Rasterzeile des Cursor (maximal 12)
	5 und 6	0	0	0	0	Bit 6 = 0 und Bit 5 = 0: Kein Blinken Bit 6 = 0 und Bit 5 = 1: Kein Cursor Bit 6 = 1 und Bit 5 = 0: Blinken Bit 6 = 1 und Bit 5 = 1: Langsam blinkender Cursor
11	0 bis 3	12	7	7	7	Letzte Rasterzeile des Cursors (maximal 12)
12	0 bis 5	–	–	–	–	Bit 8 bis Bit 13 der Startadresse des Bildschirmspeichers aus der Sicht des Controllers
	6 und 7	0	0	0	0	–
13	0 bis 7	–	–	–	–	Bit 0 bis Bit 7 der Startadresse des Bildschirmspeichers aus der Sicht des Controllers
14	0 bis 5	–	–	–	–	Bit 8 bis Bit 13 der Adresse des Cursors (maximal 1999)
	6 und 7	0	0	0	0	–
15	0 bis 7	–	–	–	–	Bit 0 bis Bit 7 der Adresse des Cursors (maximal 1999)
16	0 bis 5	–	–	–	–	Bit 8 bis Bit D der Bildschirmadresse, bei der ein Lichtgriffelimpuls auftrat
	6 und 7	0	0	0	0	–
17	0 bis 7	–	–	–	–	Bit 0 bis Bit 7 der Bildschirmadresse, bei der ein Lichtgriffelimpuls auftrat

Tabelle 2. Die 18 Register des Video-Controllers 6845 erlauben eine vielfältige Manipulation des Bildes

den Monitor um. Arbeitet der PC im Textmodus, so übergibt der Video-Controller die eingelesenen Bytes aus dem Bildschirmspeicher als ASCII-Werte an den Zeichengenerator.

Jeder ASCII-Wert gibt eine Adresse an, unter der im Zeichengenerator die Zeichenmatrix zu dem entsprechenden ASCII-Zeichen abgelegt ist. Diese Zeichenmatrix liest der Video-Controller ein und erzeugt daraus über ein sehr schnelles Schieberegister das serielle Videosignal.

Befindet sich der PC dagegen im Grafikmodus, so kann der Video-Controller die aus dem Bildschirmspeicher eingelesenen Daten direkt über das Schieberegister in das serielle Videosignal umformen.

Doch das Videosignal allein reicht nicht aus, um ein Bild auf dem angeschlossenen Monitor entstehen zu lassen. Vielmehr muß der Video-Controller auch zwei Steuersignale erzeugen, die den Bildaufbau regeln. Das eine Steuersignal ist die Bildwechselfrequenz (Vertikalfrequenz), die für die vertikale Synchronisation zuständig ist, das andere die Zeilenfrequenz (Horizontalfrequenz), die die horizontale Synchronisation übernimmt.

Die Bildröhre eines Monitors baut das Bild aus horizontalen Zeilen auf, indem die Bildröhre den Elektronenstrahl von links nach rechts auf den Bildschirm schreibt.

Beim Rücklauf des Strahls vom Zeilenende zum Anfang der nächsten Zeile wird der Strahl dunkel geschaltet (ausgetastet), damit er unsichtbar bleibt.

Der Monitor schreibt die einzelnen Zeilen von oben nach unten auf den Bildschirm und kehrt am Ende der letzten Zeile wieder diagonal zum Anfang der ersten Zeile zurück. Auch bei diesem Rücklauf wird der Strahl ausgetastet.

Diese Methode des Bildaufbaus auf dem Bildschirm ist bei allen Monitoren identisch.

Die Grafikkarten des PC stellen die Bilder deshalb mit einer echten Bildwechselfrequenz von 50 Hz (Monochromadapter und Hercules-Grafikkarte), 60 Hz (CGA- und EGA-Karte) oder sogar 70 Hz (VGA-Karte) dar.

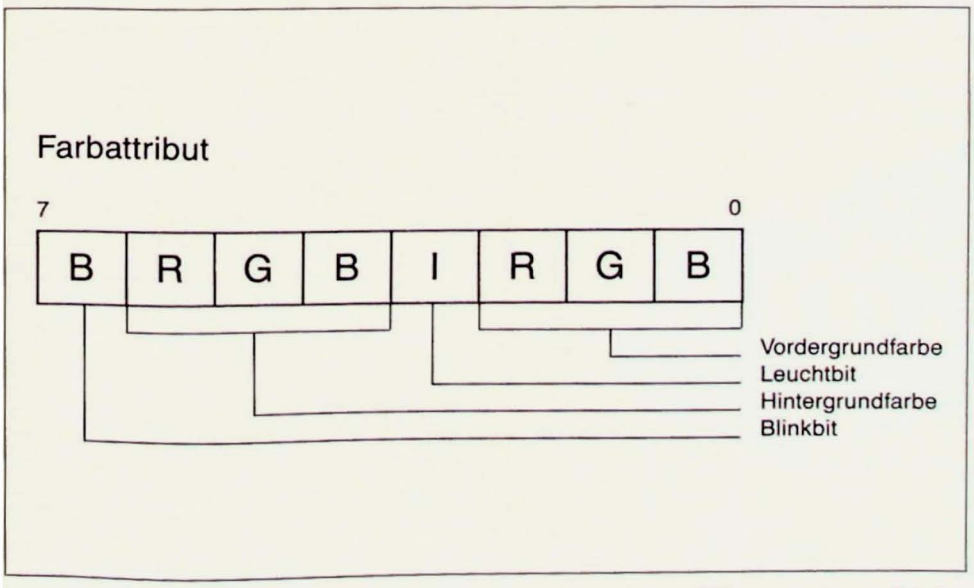


Bild 2. Das Farbattribut erlaubt in den Textmodi einer Farbgrafikkarte 256 verschiedene Kombinationen von Vorder- und Hintergrundfarbe

Der Video-Controller 6845 läßt sich über seine 18 internen Register (0 bis 17) direkt programmieren. Bei der Hercules-Grafikkarte ist diese Art der Programmierung sogar zwingend notwendig, weil das BIOS des PC diese Grafikkarte nicht kennt und damit auch nicht speziell unterstützt.

Bei der direkten Registerprogrammierung ist es sehr wichtig, daß Sie vorsichtig zu Werke gehen. Ändern Sie die Standardwerte nur in kleinen Schritten, denn bei der Programmierung von gegenüber den Voreinstellungen stark abweichenden Werten kann Ihr Monitor Schaden nehmen.

Die internen Register des Video-Controllers 6845 lassen sich über ein Adreß- und ein Datenregister ansprechen. Bei der CGA-Karte wird das Adreßregister über die Portadresse 3D4hex und das Datenregister über die Adresse 3D5hex angesprochen. Beim Monochromadapter und der Hercules-Grafikkarte lauten die Portadressen 3B4hex (Adreßregister) und 3B5hex (Datenregister).

Die 18 internen Register des Video-Controller vom Typ 6845 lassen sich über ein Adreß- und Datenregister programmieren

Um ein bestimmtes internes Register des Video-Controller mit einem Wert zu programmieren, müssen Sie zuerst die Nummer des gewünschten Registers in das Adreßregister schreiben und darauf den zu programmierenden Wert an das Datenregister übergeben. Tabelle 2 zeigt den kompletten Registersatz des 6845 mit den Voreinstellungen für verschiedene Grafikkarten und den Bedeutungen der Register.

Für erste Experimente sind speziell die Register 10, 11, 14 und 15 geeignet, weil Sie dabei keine Beschädigungen riskieren.

Die Register 10 und 11 geben die Start- und Endzeile des Bildschirm-Cursor an und bestimmen auf diese Weise das Erscheinungsbild des Cursor. Bei einem Monochromadapter und einer Hercules-Grafikkarte beginnt der Cursor bei Zeile 11 und endet bei Zeile 12. Der Cursor einer CGA-Karte verwendet die Startzeile 6 und die Endzeile 7.

Dieser Standard-Cursor belegt jeweils die unteren zwei Zeilen innerhalb einer Zeichenposition. Wenn Sie die Startzeile dagegen auf 0 setzen, so erhalten Sie einen Block-Cursor, der eine komplette Zeichenposition ausfüllt. Die Blinkfrequenz des Cursor läßt sich über die Software leider nicht verändern, weil deren Steuerung durch die Hardware erfolgt.

Die internen Register 14 und 15 des Video-Controllers bestimmen die Cursor-Position auf dem Bildschirm. Als Positionen sind die Werte 0 bis 1999 zulässig, wobei 0 die linke obere Bildschirmecke und 1999 die rechte untere Bildschirmecke bezeichnet.

Neben dem Adreß- und Datenregister, die den Zugriff auf die 18 internen Register des Video-Controllers gestatten, befinden sich auf den Grafikkarten noch weitere Register, die vom Kartentyp abhängig sind. So arbeiten beispielsweise der Monochromadapter sowie die Hercules- und CGA-Karte zusätzlich mit einem Steuerregister, das sich über die Portadresse 3D8hex (CGA-Farbgrafikkarte) beziehungsweise 3B8hex (Monochromadapter und Hercules-Grafikkarte) ansprechen läßt, und einem Statusregister, auf das Sie über die Portadresse 3D8hex (CGA-Karte) beziehungsweise 3B8hex (Monochromadapter und Hercules-Karte) zugreifen.

Unter Verwendung des Bit 0 im Statusregister können Sie beispielsweise direkt auf den Bildschirmspeicher zugreifen, ohne daß störendes »Schneetreiben« auf dem Bildschirm erscheint. Ist dieses Bit auf 1 gesetzt, so dürfen Sie ruhigen Gewissens ein Byte in den Bildschirmspeicher schreiben oder aus ihm lesen, weil in diesem Moment der horizontale Strahlrücklauf aktiv ist und der Video-Controller nicht auf den Bildschirmspeicher zugreift. Kollisionen mit dem Prozessor und die damit verbundenen Bildstörungen sind somit ausgeschlossen.

Dieses Problem tritt allerdings nur bei einer CGA-Karte auf, denn der Monochromadapter und die Hercules-Grafikkarte räumen dem Mikroprozessor bei Zugriffen auf den Bildschirmspeicher grundsätzlich Vorrang ein.

Wichtig ist auch das Bit 5 des Steuerregisters, denn über dieses Bit bestimmen Sie bei der CGA-Farbgrafikkarte, ob die Zeichen im Textmodus bei gesetztem Bit 7 des Attributbyte leuchtend oder blinkend dargestellt werden. Ist das Bit im Steuerregister auf 1 gesetzt, so blinken die Zeichen, wird es auf 0 zurückgesetzt, dann ist die leuchtende Darstellung angewählt.

Die Organisation des Bildschirmspeichers in den Textmodi des Monochromadapters sowie der Hercules- und CGA-Karte ist recht einfach zu verstehen, denn jede Zeichenposition auf dem Bildschirm entspricht zwei Byte im Bildschirmspeicher.

Das erste Byte kennzeichnet den ASCII-Wert des jeweiligen Zeichens, und das zweite Byte gibt dessen Attribut an. Tabelle 3 zeigt die Bedeutungen der Attributwerte beim Monochromadapter und der Hercules-Grafikkarte, und Bild 2 gibt Auskunft über den Aufbau des Farbattributs, das in den beiden Textmodi der CGA-Farbgrafikkarte Vorder- und Hintergrundfarbe, die Intensität der Vordergrundfarbe und die blinkende beziehungsweise leuchtende Darstellung bestimmt.

Für gewöhnliche Zwecke ist es allerdings einfacher, wenn Sie Ihre Grafikkarte nicht über die Register des Video-Controller, sondern über den Video-Interrupt 10hex programmieren, der eine Vielzahl von Funktionen bereithält, die zudem noch unabhängig von der verwendeten Grafikkarte funktionieren. Tabelle 4 listet die 16 Standardfunktionen dieses Interrupt auf. EGA- und VGA-Karten erweitern diesen Interrupt sogar noch um zusätzliche Routinen, die auf die erweiterten Fähigkeiten dieser Karten abgestimmt sind.

(Martin Hepp/ma)

Das monochrome Attributbyte	
Wert	**Bedeutung**
0	Unsichtbar
1	Unterstrichen
7	Normale Darstellung
9	Leuchtend und unterstrichen
15	Leuchtend
112	Invertiert
129	Blinkend und unterstrichen
135	Blinkend
137	Blinkend, leuchtend und unterstrichen
143	Blinkend und leuchtend
240	Blinkend und invertiert

Tabelle 3. Der Aufbau des Attributbyte im Bildschirmspeicher einer monochromen Grafikkarte

Die Funktionen des Video-Interrupt 10hex	
Funktion	**Bedeutung**
00hex	Videomodus festlegen
01hex	Cursor-Größe festlegen
02hex	Cursor positionieren
03hex	Cursor-Position lesen
04hex	Lichtgriffelposition lesen
05hex	Aktive Bildschirmseite auswählen
06hex	Bildschirm aufwärts scrollen
07hex	Bildschirm abwärts scrollen
08hex	Zeichen und Attribut von Cursor-Position lesen
09hex	Zeichen und Attribut auf Cursor-Position schreiben
0Ahex	Zeichen auf Cursor-Position schreiben
0Bhex	Farbpalette oder Farbe auswählen
0Chex	Grafikpunkt setzen
0Dhex	Grafikpunkt lesen
0Ehex	Zeichen schreiben und Cursor bewegen
0Fhex	Videomodus ermitteln

Tabelle 4. Die 16 Funktionen des Video-Interrupt 10hex

ABONNEMENT

DIE QUAL DER WAHL

ABONNEMENTPREISE
MIT VORTEIL:

DOS International:
Im Inland bzw. West-Berlin
6 Ausg. = 42,–
12 Ausg. = 82,50

Im europäischen Ausland
6 Ausg. = 55,–
12 Ausg. = 110,–

Im außereuropäischen Ausland
6 Ausg. = 75,–
12 Ausg. = 150,–

DOS Extra
Im Inland bzw. West-Berlin
4 Ausgaben 60,– DM

Im europäischen Ausland
4 Ausgaben 70,– DM

Im außereuropäischen Ausland
4 Ausgaben 80,– DM

DOS INTERNATIONAL

oder

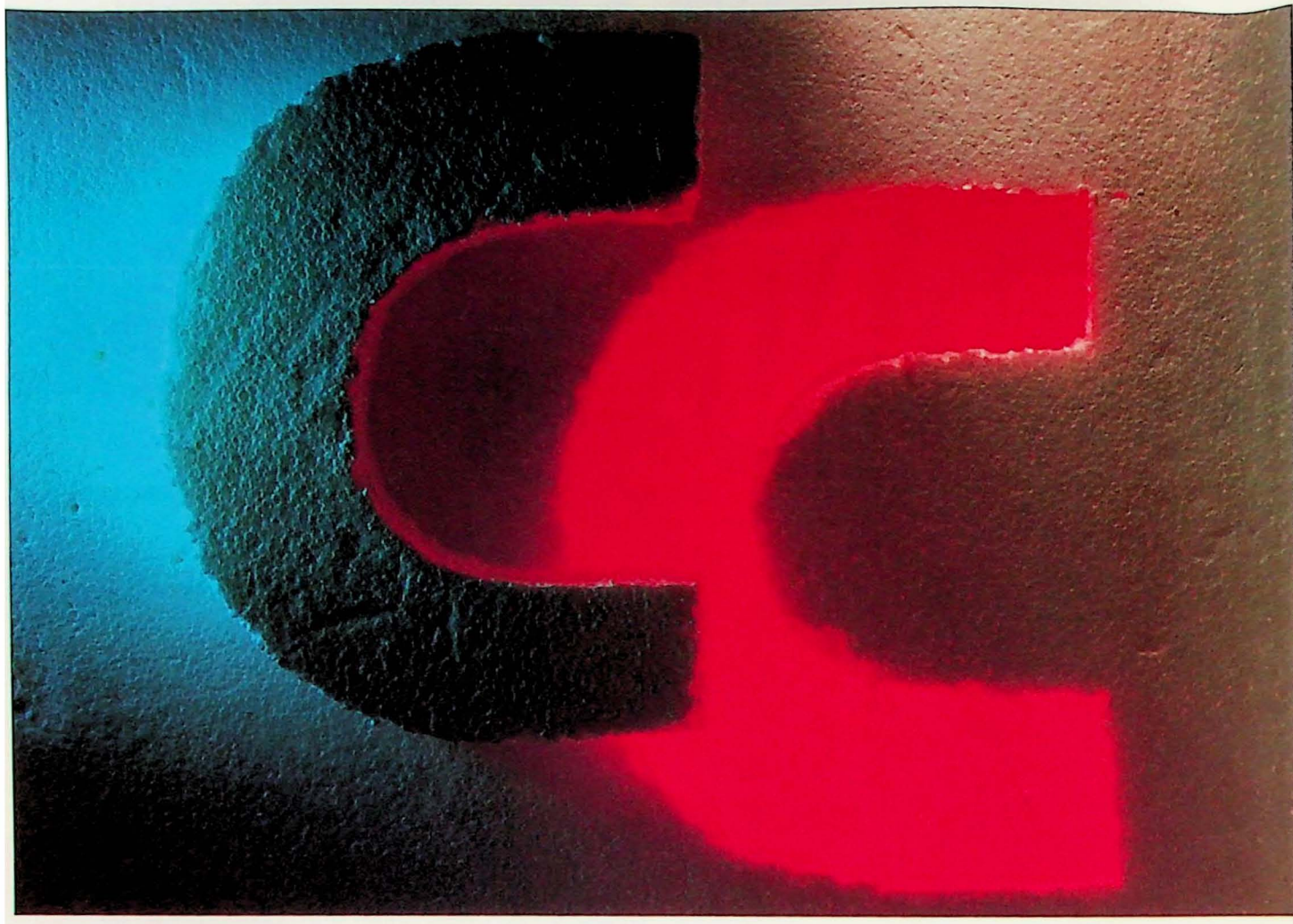
DOS EXTRA

oder beide

Widerrufsrecht: Wir garantieren jedem Abonnenten das Recht, seine Bestellung innerhalb einer Woche nach Abschluß schriftlich zu widerrufen. Die rechtzeitige Absendung des Widerrufsschreibens genügt zur Fristwahrung.

Bitte Bestellkarte benutzen

DMV-VERLAG · POSTFACH 250 · 3440 Eschwege



Die Entwicklungsgeschichte von C

Seit dem Erscheinen von leistungsfähigen und preiswerten C-Compilern, wie beispielsweise Quick C oder Turbo C, wird diese Programmiersprache immer beliebter. Anwendungsgebiete von C sind sehr vielfältig und reichen von Betriebssystemen über Datenbanksysteme bis hin zu integrierten Anwendungspaketen.

C ist eine noch relativ junge Programmiersprache – Anfang der siebziger Jahre wurde sie von Dennis M. Ritchie entwickelt. Er arbeitete in den Bell-Laboratories in den USA und stand vor dem Problem, das komplette Betriebssystem Unix in Assembler zu schreiben.

Maschinensprache war bis zu diesem Zeitpunkt der einzige Weg, um Betriebssysteme zu entwickeln. Doch Ritchie waren die Nachteile von Assemblercode ebenso klar: Maschinenprogramme lassen sich unmöglich auf andere Mikroprozessoren übertragen, selbst die Adaption auf andere Computer mit dem gleichen Prozessor führt zu kaum überwindbaren Schwierigkeiten. Ein weiterer gravierender Nachteil von Assembler ist der nahezu unerträgliche Wartungsaufwand. Von einer Codelänge von mehreren KByte an lassen sich Maschinenprogramme kaum noch lesen, geschweige denn überarbeiten.

Alle anderen auf dem Markt verfügbaren Programmiersprachen wie Algol, Fortran oder Cobol erwiesen sich als unbrauchbar. Es blieb Ritchie deshalb nichts anderes übrig, als eine neue Sprache zu entwickeln. Am Ende dieser Entwicklung stand dann C in der Form, in der wir es heute kennen. Und Ritchie hat tatsächlich das Meisterstück vollbracht, etwa 95 Prozent von Unix in C zu programmieren. Nur bei den restlichen fünf Prozent mußte er auf Assembler zurückgreifen.
Natürlich hat sich Ritchie nicht alles selbst »aus den Fingern gesogen«. Er griff selbstverständlich auf ältere Sprachen zurück. Die geistigen Väter von C sind CPL (Combined Programming Language), BCPL (Basic Combined Programming Language) und B.

Bereits in den Anfängen hatte die Portabilität die oberste Priorität

Eine der wichtigsten Eigenschaften von C ist zweifellos die Portabilität (Übertragbarkeit). Wer von einem Computer auf einen anderen umsteigt, kann praktisch die gesamte C-Software mitnehmen. Selbst wenn Sie sich für einen neuen leistungsfähigeren Compiler entscheiden, sind die Aufwendungen für eventuelle Änderungen gleich null. Diese Kom-

patibilität der einzelnen C-Versionen wird von keiner anderen Sprache erreicht, nicht einmal vom vielgerühmten Pascal: Versuchen Sie einmal, ein Programm, das extensiv von den Möglichkeiten von Turbo Pascal Gebrauch macht, auf MS-Pascal oder BCI-Pascal zu übertragen – ein aussichtsloses Unterfangen – gar nicht zu reden von den Hunderten von Basic-Dialekten.

Was für Hobby-Programmierer gilt, hat natürlich noch in viel stärkerem Maße Gültigkeit für Software-Firmen, die Programme kommerziell vertreiben. Ein großer Teil der Produktionskosten für die Software läßt sich einsparen, wenn Programme in kürzester Zeit auf anderen Computern zum Laufen gebracht werden können. Die Software-Entwicklung wird damit von den Fortschritten der Hardware unabhängiger. Aber auch hinsichtlich der eigentlichen Programmierung weist C einige Vorteile auf. Kontrollstrukturen oder Schleifenvereinbarungen stehen mit Pascal, dem »Musterschüler« der strukturierten Programmierung, durchaus auf einer Stufe: Pascals »case« heißt in C »switch-case«, die Repeat-Until- und While-Schleifen nennen sich »do-while« und »while«. Auch die allseits beliebte For-Schleife findet sich – wenn auch mit geänderter Syntax – in C wieder. Andererseits hat der Programmierer in C alle Möglichkeiten, auch auf der untersten Hardware-Ebene direkt mit den Peripheriegeräten zu kommunizieren.

C gestattet die Kommunikation auf unterster Hardware-Ebene

Dies geht so weit, daß Sie in der Lage sind, eigene Gerätetreiber, Bildschirmsteuerungen oder selbst komplette Betriebssysteme (siehe Unix) selbst zu schreiben.

Derart motiviert, können Sie sich sicher nicht mehr der Faszination entziehen, die von C ausgeht. Es stellt sich nur noch ein »kleines« Problem, bevor Sie mit dem Programmieren beginnen können: Sie brauchen einen Compiler. Für MS-DOS-Computer gibt es schon beinahe mehr C- als Pascal-Compiler. Das Preisspektrum reicht von etwa 100 Mark für das Zorlite-Sparpaket über Turbo C für zirka 350 Mark bis hin zum professionellen Entwicklungssystem MS-C 5.0 oder Lattice C für über 1500 Mark.

Neben einem Compiler ist es durchaus auch ratsam, weitergehende Literatur zur Unterstützung anzuschaffen. Den absoluten Klassiker im C-Sektor stellt das Standardwerk »The C Programming Language« dar. Dieses vielgelobte Werk wurde von den Entwicklern der Programmiersprache C (Ritchie/Kernighan) geschrieben. In Fachkreisen spricht man von diesem Buch als der »C-Bibel«. Eine deutsche Übersetzung (»Programmieren in C«) bietet der Hanser-Verlag an.

Doch nun zurück zur Programmiersprache. C ist eine modulare Sprache, die mit einem sehr kleinen Befehlsvorrat auskommt. Insgesamt definiert der ANSI-Standard nur 32 Schlüsselworte, die Sie der Tabelle entnehmen können. Ein C-Programm besteht aus mehreren einzelnen Funktionen, die jeweils eine bestimmte Aufgabe erledigen. Unter Funktionen versteht man das, was man in Basic mit »gosub« oder »print fnx(y)« aufruft und was in Pascal mit »funktion« oder »procedure« bezeichnet wird. C macht dabei keinen Unterschied zwischen Funktionen und Unterprogrammen.

Ein kleiner Befehlsvorrat erleichtert den Einstieg in die Programmiersprache C

Jedes C-Programm besteht aus mindestens einer Funktion. Die Hauptfunktion erhält den Namen »main« und wird beim Programmstart automatisch aufgerufen. »main« kann dann von sich aus die anderen Funktionen – sofern vorhanden – anspringen. Beim Funktionsaufruf werden in Klammern Variablen als Argumente übergeben. Da das in diesem Fall unnötig ist, müssen Sie dem Funktionsnamen leere Klammern folgen lassen wie beispielsweise »main()«.

Es gibt einen sehr wichtigen Punkt, den Sie bei der Eingabe der Programme beachten sollten: Ein ungeschriebenes Gesetz besagt, daß C-Programme generell in Kleinbuchstaben geschrieben werden. Die einzige Ausnahme stellen in diesem Fall Konstanten dar – insbesondere #Define-Angaben. Während die Schreibweise den meisten Basic-Interpretern vollkommen gleichgültig ist, unterscheidet ein Großteil der C-Compiler genau zwischen Namen wie »main«, »MAIN« und »Main«! Diese Konventionen sind jedoch in der C-Sprache die reine Ausnahme. Im Programmaufbau sind Ihrer Kreativität kaum Grenzen gesetzt. Beachten Sie nachstehende Beispiele:

```
main()
{
printf("DOS-Extra");
}
```

Dieses Programm gibt den Text »DOS-Extra« auf dem Bildschirm aus. Den gleichen Effekt erreichen Sie auch mit:

```
main(){printf
("DOS-Extra")
;}
```

Datentypen:	static
char	typedef
double	**Anweisungen:**
float	break
int	case
long	continue
unsigned	default
Abgeleitete Datentypen:	do
	else
enum	for
struct	goto
union	if
Parametertypen:	switch
const	while
volatile	**Unterprogramme:**
Speicherklassen:	return
auto	void
extern	**Speicherverwaltung:**
register	sizeof

Tabelle. Der ANSI-Standard definiert für die Sprache C lediglich 32 Schlüsselworte

Der Compiler also versteht Programme in jedem Bildschirmformat, ist dies bei Ihnen jedoch auch der Fall? Denken Sie daran: Sie sollten Ihre Programme auch dann noch verstehen können, wenn Sie sie erst nach einigen Monaten wieder betrachten. Im Demo-Programm wurde die Ausgabefunktion »printf« bereits vorgestellt. Zusätzlich werden mit jedem Compiler verschiedene Routinen mitgeliefert. So enthalten unter anderem die Produkte von Microsoft zirka 300 C-Funktionen aus den unterschiedlichsten Bereichen. So sind etwa Funktionen für Ein-/Ausgabe, mathematische Berechnungen oder String-Manipulationen vorhanden.

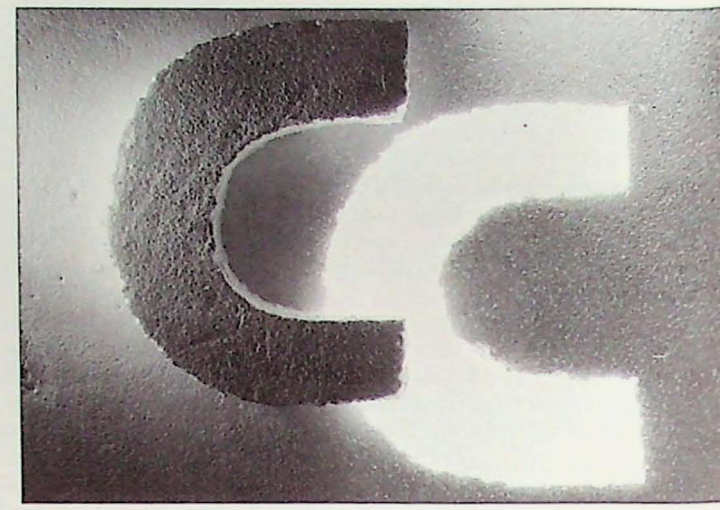
Bei der Programmierung in C ist Disziplin gefordert

Ein kleiner Nachteil ergibt sich aus der Tatsache, daß C eine sehr maschinennahe Sprache ist. Compiler und Laufzeitbibliothek (das ist der Code, der zu jedem übersetzten Programm hinzugebunden wird, damit es ablaufen kann) entdecken normalerweise keine Über- oder Unterschreitungen der Wertebereiche von Variablen. Die Programme schneiden einfach den Teil der Zahl weg, den sie nicht mehr verwerten können. Wenn Sie zum Beispiel als Ergebnis einer ganzzahligen Rechnung mit Integervariablen (maximal zwei Byte) eine 20 Bit breite Zahl erhalten, »vergißt« das Programm die vier vordersten Bit und läßt nur die 16 darstellbaren übrig.

Diese Eigenschaft fordert zwar vom Programmierer einige Disziplin, hat aber den Vorteil, daß die erzeugten Programme kleiner und schneller werden. Und das wiederum ist sehr erfreulich! Wer also effektive und schnelle Programme schreiben will und über die nötige Selbstdisziplin verfügt, der ist mit der Programmiersprache C gut bedient.

(Harald Krause/hi)

Die Routine »printf« unter der Lupe



Nachdem in einem Programm alle Berechnungen oder Dateizugriffe erfolgt sind, besteht zudem meist der Wunsch, die Resultate in formatierter Weise sichtbar darzustellen. Zu diesem Zweck hält die C-Bibliothek mehrere Routinen bereit. Die leistungsfähigste Funktion nennt sich »printf«.

Sicherlich kennen Sie das Problem, daß nach erfolgter Berechnung komplizierter Formeln die Ergebnisse in übersichtlicher Weise am Bildschirm oder Drucker ausgegeben werden sollen. Gleichgültig, ob hierbei eine spezielle Tabellenform oder sonstige Formatierungen, wie zum Beispiel links- oder rechtsbündige Ausgaben, berücksichtigt werden sollen – es sind meist umfangreiche Manipulationen erforderlich. Da C selbst nur über einen sehr kleinen Befehlsvorrat verfügt, bemerkt man sehr schnell das Fehlen spezieller Ein- und Ausgabekommandos.

Die eigentlichen C-Befehle dienen vorrangig der Steuerung des Programmflusses. Die einzige Möglichkeit, Ausgaben vorzunehmen, stellen die mitgelieferten Funktionen in den Bibliotheksdateien mit dem Namen »stdio.h« oder »conio.h« dar. Diese Abkürzungen stehen für »Standard Input/Output« (Standard Ein- und Ausgabe) beziehungsweise »Consol Input/Output« (Ein-/Ausgabe auf Konsole). Hierin enthalten ist unter anderem auch eine spezielle Funktion namens »printf« oder eines der verschiedenen Derivate (siehe Tabellen 2 und 3).

Überraschenderweise stimmen die Funktionen zwischen den C-Bibliotheken verschiedener Compiler überein. Sie orientieren sich fast alle an der Unix-Bibliothek von Kernighan und Ritchie. Aus diesem Grunde können Sie die Printf-Funktion problemlos extensiv nutzen, ohne die Portabilität Ihres Programms einzuschränken.

Doch bevor die Arbeitsweise näher beschrieben wird, ist es notwendig, das allgemeine Format kennenzulernen. Es lautet:

```
printf(Format[,Argument]...);
```

Das Format steht hierbei für die Zeichenkette, die bestimmte Formatierungsangaben beinhaltet. Im einfachsten Falle wird lediglich ein beliebiger Text eingetragen. Vergleichen Sie hierzu das nachfolgende Beispiel:

```
main()
{printf("DOS-Extra");
}
```

Ergebnis dieser Funktion ist die Ausgabe des Textes »DOS-Extra« auf dem Bildschirm beziehungsweise am Standardausgabekanal. Dieses Resultat erreicht man jedoch auf schnellerem Wege über »puts«. Der hierzu notwendige Aufruf lautet:

```
main()
{
puts("DOS-Extra");
}
```

Mit der Ausgabe soll jedoch meist auch die Position angegeben werden. Aus diesem Grunde steht das Formatierungszeichen »\« (Backslash) zur Verfügung. Dieses Zeichen dient zur rein hardwarebezogenen Ausgabesteuerung. Die in der Tabelle 1 aufgezeigten Parameter werden verwendet, um beispielsweise den nächsten Tabulatorstop anzuspringen oder einen Zeilenvorschub auszulösen:

```
main()
{
printf("\n DOS \t Extra");
}
```

Steueranweisung	Bedeutung
\a	Piepston (Bell)
\b	Cursor ein Zeichen zurück
\f	Seitenvorschub (Formfeed)
\n	Zeilenvorschub (Linefeed)
\r	Wagenrücklauf (Carriage Return)
\t	horizontaler Tabulator (HTAB)
\v	vertikaler Tabulator (VTAB)
\x	ASCII-Zeichen mit dem Wert der folgenden Hexadezimalzahl (Beispiel: \x0D = Wagenrücklauf)
\O	ASCII-Zeichen mit dem Wert der folgenden Oktalzahl (Beispiel: \O15 = Wagenrücklauf)
\\	Rückstrich »\«(Backslash)
\'	Hochkomma »'«
\"	Anführungszeichen »"«
\?	Fragezeichen »?«

Tabelle 1. Steuercodes der Printf-Funktion

Formatanweisung	Bedeutung
%c	Textzeichen vom Typ »char«
%d	vorzeichenbehafteter ganzzahliger Wert vom Typ »signed int«
%e	vorzeichenbehafteter Fließkommawert in Exponentendarstellung mit zusätzlicher Formatierungsmöglichkeit
%E	vorzeichenbehafteter Fließkommawert in Exponentendarstellung mit zusätzlicher Formatierungsmöglichkeit
%f	Fließkommazahl in Festkommadarstellung vom Typ »double«
%g	Fließkommazahl vom Typ »double«
%i	vorzeichenbehafteter ganzzahliger Wert vom Typ »signed int«
%n	Zeiger auf Integer-Wert (Pointer)
%o	vorzeichenloser Oktalwert
%p	Zeiger (far pointer) im Format Segment:Offset
%s	Zeichenkette (String)
%x	vorzeichenloser ganzzahliger Hexadezimalwert, wobei Alpha-Zeichen in Kleinbuchstaben
%X	vorzeichenloser ganzzahliger Hexadezimalwert, wobei Alpha-Zeichen in Großbuchstaben

Tabelle 2.Die verschiedenen Varianten von Printf

Nach Ablauf des Programms erhalten Sie den Text »DOS« in einer neuen Zeile »\n« sowie die Zeichenkette »Extra« durch einen Tabulatorsprung getrennt. Andere Steuercodes wie etwa »\f« (Formfeed) sind speziell bei der Druckausgabe nützlich.
Interessant wird die Printf-Funktion aber dann, wenn Daten in einem besonderen Format ausgegeben werden sollen. Um der Routine mitzuteilen, welche Art der Formatierung durchzuführen ist, sind zwei Steuerzeichen reserviert. Zum einen handelt es sich hierbei um das Prozentzeichen »%« und zum anderen um den Rückstrich »\« oder Backslash.
Das Prozentzeichen hat die Aufgabe, Formatierungen von Variablen zu definieren. Alle vorhandenen Steuerungsmöglichkeiten können Sie Tabelle 1 entnehmen. Angenommen der Inhalt einer Variablen namens »test« soll auf dem Bildschirm ausgegeben werden.

```
char test[]="DOS-Extra";
printf("%s",test);
```

Dieser Programmausschnitt bewirkt, daß anstelle der Zeichenfolge »%s« der Inhalt der Variablen angezeigt wird. Wird dagegen ein numerischer Wert aufbereitet, ist der Formatbeschreiber entsprechend zu ändern:

```
int test=23;
printf("%d",test);
```

Gleichzeitig mit der Ausgabe einer Variablen kann selbstverständlich auch ein konstanter Textteil gezeigt werden:

```
printf("Ausgabe: %d",test);
```

Dies hat zur Folge, daß auf dem Bildschirm die Anzeige »Ausgabe: 23« erscheint.
Aber die Funktion kann noch viel mehr. Sie rechnet Zahlen in verschiedene Zahlensysteme um und druckt diese dann formatiert aus. Dies kann zum Beispiel folgendermaßen aussehen:

```
main()
{
printf("%x",32767);
}
```

»%x« veranlaßt »printf« dazu, den dezimalen Wert 32767 in die hexadezimale Darstellung 7FFFhex umzurechnen. Genauso, wie Sie Konstanten und Variablen mischen können, besteht auch die Möglichkeit, mehrere Variablen in unterschiedlicher Aufbereitung anzuzeigen. Allerdings muß dabei die Anzahl der Formatkenner mit den angegebenen Variablen übereinstimmen. Gibt es da Unterschiede, macht der Computer meistens »Unsinn«, und die Resultate fallen mitunter recht amüsant aus.

```
main()
{
int test=10;
printf("%x %d %o",test,test,test);
}
```

Funktionsname	Parameter	Aufgabe	Rückgabewerte
fprintf	1. FILE *Stream 2. char *Formatstring 3. Argumente	die Ausgabe erscheint nicht auf dem Bildschirm, Ausgabe erfolgt auf »STREAM«	
printf	1. char *Formatstring 2. Argumente	formatierte Ausgabe erscheint auf dem Standardausgabekanal (meist Bildschirm)	
sprintf	1. char *Zeichenkette 2. char *Formatstring 3. Argumente	die Ausgabe wird in die »Zeichenkette« übertragen	
vfprintf	1. FILE *Stream 2. char *Formatstring 3. va_list Parameter	hierbei ist keine feste Angabe der auszugegebenden Werte nötig; variable Werte werden über Liste vom Typ »va_list«;Ausgabe erfolgt auf »STREAM«	
vprintf	1. char *Formatstring 2. va_list Parameter	wie »vfprintf«, jedoch erfolgt die Ausgabe auf Standardausgabe (meist Bildschirm)	
vsprintf	1. char *Zeichenkette 2. char *Formatstring 3. va_list Parameter	wie »vfprintf«, jedoch wird die formatierte Ausgabe in »Zeichenkette« übertragen	
cprintf	1. char *Format 2. Argumente	Arbeitet analog zu »printf«, die Ausgabe erfolgt jedoch nur auf dem Bildschirm	

Tabelle 3. Die verschiedenen Printf-Derivate

-	Links-Justierung, sofern Feldbreite angegeben wird; Standard: Rechts-Justierung
+	setzt das jeweile Vorzeichen (sofern Signed-Variable) der Ausgabe voran; Standard: Vorzeichen nur bei negativen Werten
Leer	Stellt ein Leerfeld bei positiven Werten voran (nur bei Signed-Variablen); Standard: ohne Leerzeichen

Tabelle 4. Erweiterte Möglichkeiten der Formatierung

Das obige Programm bewirkt, daß die Variable »test« zuerst als Hexadezimalwert, anschließend als Dezimalwert und zuletzt als Oktalwert ausgegeben wird.

Die Umrechnung in andere Zahlensysteme ist überhaupt kein Problem

Neben diesen bisher genannten unterschiedlichen Varianten lassen sich weitere Spezifikationen angeben. Sie können beispielsweise

- die Ausgabelänge
- die Genauigkeit
- den Vorzeichenausweis
- die Justierung

vereinbaren. Hierbei gilt das folgende allgemeine Format:

```
%[Vorzeichen][Weite][.Genauigkeit]type
```

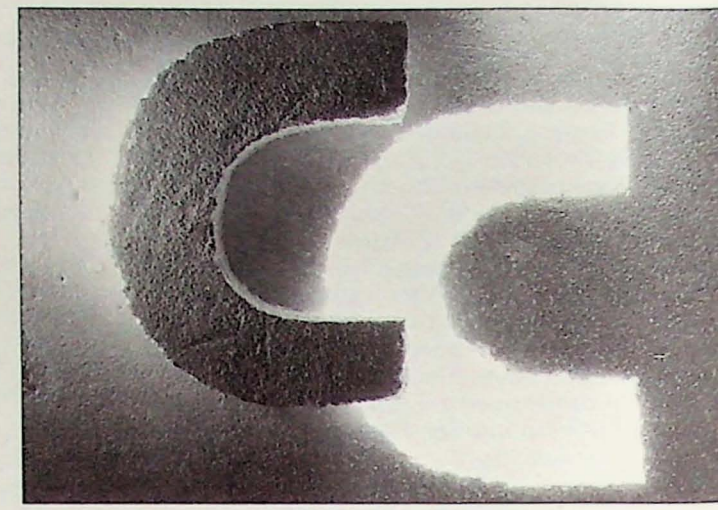
Alle Einstellungsvarianten werden aus Tabelle 4 ersichtlich. Am besten lassen sich diese erweiterten Möglichkeiten anhand eines Beispiels verdeutlichen. Um bei einer tabellarischen Gliederung alle Werte kommagerecht untereinander aufzuführen, kann folgender Printf-Aufruf gewählt werden:

```
main()
{
float a=76,874;
float b=2,23;
float c=134,0;
printf("\n%10.3f",a);
printf("\n%10.3f",b);
printf("\n%10.3f",c);
}
```

Dieses Programm setzt eine Gesamtbreite von zehn Zeichen pro Variable, wobei jeweils drei Nachkommastellen aufbereitet werden. Ist es aus irgendwelchen Gründen erforderlich, den linken freien Bereich durch Null-Werte zu ersetzen, so erreichen Sie dies durch die Angabe »printf("\n%010.3f",a)«.
Wie Sie sehen, verbirgt sich hinter der Printf-Routine ein sehr vielseitiges Werkzeug. Weitere Möglichkeiten der formatierten Ausgabe stellen die Funktionen aus Tabelle 3 dar. So hat zum Beispiel die Routine »sprintf« den Zweck, die Ausgabe in einer Zeichenkette einzutragen, anstatt diese auf dem Bildschirm auszugeben. Grundsätzlich entspricht sie jedoch der allgemeinen Printf-Funktion.

(Harald Krause/hi)

Die Macht der Operatoren



In der Programmiersprache C besitzen die Operatoren einen besonders hohen Stellenwert. Vergleicht man die Anzahl der Schlüsselworte mit den vorhandenen Operatoren, so zeigt sich, in welchem Maße sich C der Operatoren bedient.

Die Anzahl der Operatoren, die C besitzt, wird wohl kaum von einer anderen Programmiersprache übertroffen. Eine vollständige Auflistung finden Sie in Tabelle 1. Beachten Sie hier unbedingt die jeweiligen Hierarchiestufen. Wie aus der Mathematik bekannt, ist es ein immenser Unterschied, ob die Schreibweise x = a*b + 3 oder aber x = a*(b + 3) lautet. Tabelle 1 ist deshalb so aufgebaut, daß die höchste Rangstufe ganz oben und die niedrigste ganz unten steht.

Die Angabe der Assoziativität (Verbindung) gibt Auskunft über die Rangfolge von Operatoren auf gleicher Ebene. Wie Sie anhand des Operators » – « (Minus) sehen können, befindet sich das Zeichen »*« (Multiplikation) auf einer höheren Hierarchiestufe. Die Aussage, daß der Additionsoperator » + « in einer Links-Rechts-Assoziativität steht, bewirkt, daß zuerst die Operation auf der linken Seite ausgeführt wird. Soll beispielsweise x = a-b + 3 errechnet werden, so wird zuerst der Wert der Variablen »a« um den Inhalt der Variablen »b« vermindert und anschließend der Wert 3 zu dem Ergebnis addiert.

Die Rangfolge der Operatoren spielt eine wichtige Rolle

Die Bedeutung der einzelnen Operatoren finden Sie in der Tabelle 2. Verschiedene Assembler-Befehle wie »inc« und »dec« wurden direkt in C übernommen. Diese Befehle wurden jedoch zu Operatoren umfunktionert. Die Inkrement-Anweisung (+ +) hat die Aufgabe, den Wert einer Variablen um 1 zu erhöhen. Der Dekrement-Operator (–) hingegen vermindert den Wert um 1. Daß hier nur Integer-Zahlen (Ganzzahlen) benutzt werden können, liegt im Wesen des In- beziehungsweise Dekrement-Befehls.

```
main(){
int i=3;
i++;
printf("%d",i);
}
```

Zu Beginn des obigen Programms wurde die Variable »i« mit dem Wert 3 vorbelegt. Nach Ausführung der Inkrement-Anweisung erhalten Sie als Ausgabe die Zahl 4 am Bildschirm. Das gleiche Ergebnis liefert selbstverständlich auch »i = i + 1« oder »i + = 1«. Allerdings ist der Zeitaufwand hier erheblich höher.

Diese beiden Operatoren beinhalten aber noch mehr. Hier kommt es speziell auf die Stellung in der Anweisung an. Betrachten Sie folgendes Programm:

```
main(){
int x=3;
int y=3;
printf("\t%d",x++);
printf("\t%d",++y);
printf("\t%d %d ",x,y);
}
```

Im Rahmen der ersten Printf-Funktion wurde die Variable »x« inkrementiert. Der Operator » + + « nimmt hierbei die sogenannte Postfix-Position ein, steht also dahinter. Eine kleine Eselsbrücke könnte hier die »Deutsche Bundespost« sein. Als Ausgabe erhalten Sie nun, wahrscheinlich etwas überraschend, den bisherigen x-Wert 3. Denn erst nachdem x ausgegeben wurde, findet die Erhöhung statt. Die zweite Printf-Anweisung gibt dagegen sofort den auf 4 erhöhten Wert aus. Man spricht hierbei von Präfix-Schreibweise (vorrangig). Die letzte Ausgabe beweist dies.

Wie Sie sicherlich bisher schon bemerkt haben, ist das Programmieren in C eine recht kurze und prägnante Angelegenheit.

Zwar liegt ja bekanntlich in der Kürze die Würze, doch kann man es auch übertreiben. Als Resultat stellen sich dann Programme ein, die einem bereits nach wenigen Tagen Rätsel aufgeben. Als abschreckendes Beispiel dient folgende Demonstration:

```
x=++a+ ++b-c-;
```

Wer blickt hier noch durch? Schreiben Sie deshalb Ihre Programme übersichtlich, und verzichten Sie auf übertriebene Kürze. Einer guten Dokumentation sollte schon deswegen ein besonderer Stellenwert eingeräumt werden.

```
if(a==3 && b>7) ...
if(a!=3| |   c<7) ...
```

Diese Schreibweise ist wesentlich übersichtlicher als etwa:

```
if(a==3){
if(b>7) ... } /* END IF */
else { if (a!=3) {
if(c<7) ...
} /* END ELSE */
```

Obwohl in der Kürze die Würze liegt – Übersichtlichkeit geht vor

Die Programmiersprache C kennt darüber hinaus noch eine zweite Art der If-Else-Abfrage, die sogenannte bedingte Bewertung oder das Konditional. Eine kleine Demonstration verdeutlicht die Anwendung:

```
main()
{
int a=3,b=12,c;
c=(a==b)?1:11;
printf("%d",c);
}
```

Operator	Assoziativität
() [] . ->	links-rechts
! ~ + + – * &(Zeiger)	
sizeof	rechts-links
* / %	links-rechts
+ -	links-rechts
> > < <	links-rechts
> < > = < =	links-rechts
= = ! =	links-rechts
&	links-rechts
^	links-rechts
|	links-rechts
&&	links-rechts
| |	links-rechts
?:	rechts-links
= + = -= * = / =	rechts-links
, (Sequenz)	links-rechts

Tabelle 1. Übersicht aller vorhandenen Operatoren nach Kerninghan und Ritchie

	Operator	Bedeutung	Beispiel
( )	arithmetischer Vorrang und Funktionskennzeichnung Cast-Operator (explizite Typumwandlung)	a = b*(3 + 4)	main() a = (int)b
[ ]	Tabellenelement	tab[3]	
. ->	Strukturverweise	person.name	
!	Verneinung	if(!(a = = 3))...	
~	Komplement	a = ~b	
+ + −	In- und Dekrement	a + + a−	
* &(Zeiger)	Stern-und Adreßoperator	*a = &b	
sizeof	Größenoperator (liefert die Größe in Byte)	a = sizeof(b)	
* /	Multiplikation/Division	a = b*2/c	
%	Modulo (Rest)-Operator	a = 5%2	
+ -	Addition/Subtraktion	a = b + 3-c	
> > < <	Rechtsshift/Linksshift	a = 8 > > 1 und a = 8 < < 1	
> < > = < =	Vergleichsoperatoren	if(a > b)...	
= = !=	Vergleichsoperatoren (ist gleich/ungleich)	if(a! = b)...	
&	bitweise und	a = b&2	
^	exklusiv oder	a = b^2	
|	bitweise oder	a = b| 2	
&&| |	logisches und/oder	if(a = = 2 && b = = 2) ...	
? :	bedingte Bewertung	c = a > b?1:0	
=	Zuweisung	a = 2	
+ = - = * = / =	Zuweisungsverbund-Operator	a* = 3 identisch mit a = a*3	
,	Sequenzoperator	for(a = 1,b = 9;a > 3;a + + ,b- = 2)...	

Tabelle 2. Die Bedeutung und Anwendungsmöglichkeiten der C-Operatoren

In der zweiten Programmzeile findet ein Vergleich der Variablen »a« mit »b« statt. Trifft dieser Vergleich zu, ist also der Wert der Bedingung ungleich 0 (If-Zweig), so wird die Zahl 1 der Variablen »c« zugewiesen. Ist die Bedingung falsch (Else-Zweig), wird der Wert 11 übertragen. Obiges Programm ist folglich völlig identisch mit:

```
if(a==b) c=1;
else c=11;
```

Aber auch bei Auswertungen leistet das Konditional gute Dienste. Betrachten Sie hierzu folgenden Programmausschnitt:

```
/* Berechnungsende */
printf("Das Ergebnis: %d ",i);
printf("Teil%c",(i==1)?' ':'e');
```

In Abhängigkeit des Inhalts der Variablen »i« wird die Ausgabe entsprechend angepaßt.

Eine Alternative zu der If-Anweisung finden Sie im Konditional

Ihre Bildschirmausgabe könnte nun beispielsweise so aussehen:»Das Ergebnis: 3 Teile« oder »Das Ergebnis: 1 Teil«. Zwar ist es generell möglich, innerhalb der bedingten Bewertung auch Verschachtelungen und Verknüpfungen einzubauen, doch sollten Sie dies in Hinblick auf Übersichtlichkeit vermeiden.

Einige Operatoren kamen bis jetzt noch nicht zur Sprache. Speziell in Sachen »Bitmanipulation« hält C einige Feinheiten bereit. Generell sollten nur ganzahlige Variablen vom Typ »int« oder »char« verwendet werden, weil Operanden vom Typ »float« oder »double« in bezug auf ihre interne Darstellung Probleme in sich bergen. Die Vorteile der Bitoperatoren erstrecken sich von Speicherplatzoptimierung bis hin zur Geschwindigkeitssteigerung.

Besonders einfach in der Handhabung sind die sogenannten Schiebe- oder Shift-Operatoren. Hier wird unterschieden zwischen

- Links-Shift (schiebe links)
- Rechts-Shift (schiebe rechts)

Der Links-Shift-Operator wird häufig zur »schnellen« Multiplikation eingesetzt. Voraussetzung ist aber, daß die Multiplikatorwerte Zweierpotenzen darstellen. Angenommen, der Variablen »c« soll das Produkt der Multiplikation von 64x4 zugewiesen werden. Die herkömmliche Schreibweise würde dann lauten: c = 64 x 4. Das gleiche Resultat, nämlich 256, erreichen Sie aber auch durch »c = 64 < < 2«, und das in kürzerer Zeit. Wie funktioniert das?

Um dies genauer zu analysieren, ist es notwendig, die binäre Darstellungsform zu kennen. Der Wert 64 (dezimal) entspricht im Binärsystem »100 0000«. Die Zahl 4 stellt ebenso eine Zweierpotenz (2^2) dar. Schieben (shift) Sie nun den Binärwert »100 0000« um den Exponenten des Multiplikators nach links (2 Stellen) lautet das Ergebnis: »1 0000 0000«. Dies entspricht dem Dezimalwert »256«.

Entsprechend den obigen Ausführungen arbeitet der Rechts-Shift-Operator. Da der binäre Wert nach rechts geschoben wird, handelt es sich hierbei um eine »schnelle« Division. Dies ist aber nur möglich, wenn die Divisorwerte Zweierpotenzen darstellen. Das Ergebnis der Rechnung »c = 256 / 4« ist identisch mit »c = 256 > > 2«.

Wie Sie aus den Tabellen ersehen können, verfügt die Programmiersprache C noch über weitere Möglichkeiten der Bitmanipulation. Die Schreibweise ist hier besonders wichtig. Während die logischen Operatoren »| |« (logisches »oder«) und »&&« (logisches »und«) im Zusammmenhang mit Ausdrücken verwendet werden, benutzt man die Bitoperatoren »|« (bitweise »oder«) und »&« (bitweise »und«) zur Auswertung der Bitstellungen.

Die Bitmanipulation ist zwar schnell, aber sie ist auch gefährlich

Nach der Umwandlung der Werte 20 und 7 in das binäre Zahlensystem werden diese Werte miteinander »verglichen«. Das Ergebnisbit wird auf »wahr« (1) gesetzt bei der:

- bitweisen Oder-Verknüpfung (»|« = ASCII-Zeichen 124), sobald mindestens ein Bitschalter wahr (1) ist
- bitweisen Und-Verknüpfung (»&«), wenn beide Bit-Schalter wahr sind
- Exklusiv-Oder-Verknüpfung (»^« = ASCII-Zeichen 94), wenn nur ein Bitschalter von beiden wahr ist.

Im Gegensatz zu den bisherigen Operatoren wird der Komplementoperator »~« recht selten benutzt. Die Aufgabe besteht darin, jedes Bit eines Byte beziehungsweise mehrerer Bytes umzudrehen. So wird aus jeder 1 eine 0 und entsprechend aus jeder 0 eine 1.

(Harald Krause/hi)

Und

&	0	1
0	0	0
1	0	1

Oder

|	0	1
0	0	1
1	1	1

Exklusiv Oder

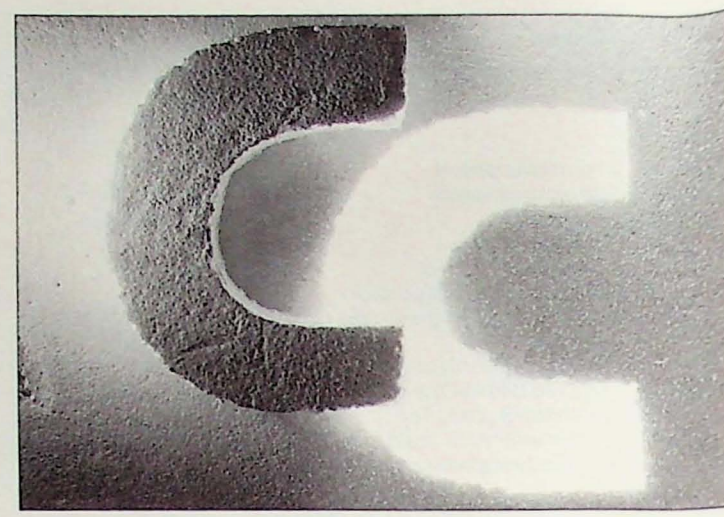
^	0	1
0	0	1
1	1	0

Komplement

~	0	1
	1	0

Die Wahrheitstabellen zu den logischen Operatoren

Zeiger sicher im Griff



Mit Hilfe der direkten Adressierung lassen sich viele Probleme hinsichtlich des Speicherplatzes lösen. Daß der Umgang mit Zeigern, den sogenannten »Pointern«, gar nicht so kompliziert ist, beweist dieser Beitrag.

Für einen Großteil der Programmierer ist der Umgang mit Zeigern und Adressen eine besonders heikle Variante der Programmentwicklung. Die Folge davon ist, daß meist nur routinierte Profis diese effiziente Art der Speicherverwaltung nutzen. Um etwas Licht ins Dunkel zu bringen, sollten Sie sich daher einmal das Konzept von Zeigern und Adressen zu Gemüte führen.

Zu Beginn ist zuerst einmal zu klären, was sich hinter einem Zeiger verbirgt. Ein Zeiger ist nichts anderes als der Verweis auf die Adresse einer Variablen oder einer Konstanten. Diese Adresse wird immer dann benötigt, wenn auf Variablen zugegriffen werden muß. Stellen Sie sich den Arbeitsspeicher eines PC als eine Bibliothek vor, so repräsentiert die Adresse nichts anderes als den Ablageplatz eines bestimmten Buches. Um dieses Buch wiederzufinden, ist es nötig, das Regal und die Bücherreihe zu kennen.

Der Arbeitsspeicher eines PC ist ähnlich strukturiert, jedoch handelt es sich dabei nicht um Bücherregale, sondern um Segment- und Offset-Adressen. Für diejenigen, die bereits über Assemblerkenntnisse verfügen, ist das sicherlich nichts Neues.

Die Technik der Zeigerverwaltung nutzt jede Programmiersprache

Selbst das simple Basic arbeitet mit Zeigern – nur sieht man sie nicht. Im Gegensatz dazu läßt C die Verwendung von Zeigern als Sprachelement zu. Dies ist zwar nichts Außergewöhnliches, da auch andere Sprachen wie Pascal oder PL/1 diese Möglichkeiten intensiv nutzen, doch reicht die Anwendung der Zeiger in C wesentlich weiter. So verfügt C beispielsweise über die Zeigerarithmetik, das heißt, das Addieren, Subtrahieren oder Vergleichen von Zeigern ist möglich. Dies ist in anderen Sprachen nicht gegeben.

Eigens für die Ermittlung von Adressen stellt C einen eigenen Operator zur Verfügung. Setzen Sie das kaufmännische »und«, also das Zeichen »&«, vor eine Variable, so erhalten Sie die Adresse. Verwechseln Sie jedoch den Adreßoperator nicht mit dem Bitoperator. Zwischen der Variablen und dem Operator darf sich kein Leerzeichen befinden. Mit Hilfe der Printf-Funktion läßt sich diese Adresse nun problemlos am Bildschirm ausgeben:

```
int wert;
printf("%u",&wert);
```

Beachten Sie hierbei, daß die Formatierung »u« (vorzeichenlose Darstellung) gewählt ist. Diese Darstellung ist nötig, da die Arbeitsspeicheradresse durchaus größer als 32767 sein kann. Bei der herkömmlichen Formatierung »%d« kann es ansonsten zu einer Fehlinterpretation kommen.

Nachdem Sie nun die Adresse einer Variablen ermittelt haben, soll als nächstes der Inhalt (also die Adresse) an eine andere Variable übergeben werden. Dies hat zur Folge, daß beide Variablen den gleichen Inhalt besitzen beziehungsweise auf den gleichen Wert zeigen. Da logischerweise nicht jede Variable eine Adresse aufnehmen kann, ist ein besonderer Variablentyp erforderlich. Die Programmiersprache C hält hierzu den Operator »*« bereit. Eine Variable wird als Zeiger erkannt, wenn vor dem Variablennamen das Zeichen »*« gesetzt ist. Dies kann in der Praxis dann folgendermaßen aussehen:

```
int *zeiger;
```

Wie unschwer zu erkennen ist, funktioniert die Vereinbarung einer Zeigervariablen ähnlich wie die einer normalen Variablen. Neben dem Namen ist immer der Variablentyp anzugeben. Der Grund liegt darin, daß die Zeigerarithmetik typgebunden ist. Doch dazu später mehr.

Die Übergabe einer Adresse kann wie folgt durchgeführt werden:

```
int wert=9;
int *zeiger;
zeiger=&wert;
```

Um die Adresse der Variablen »wert« zu ermitteln, benötigen Sie wiederum den Operator »&«.

Nach der Adreßübergabe erhalten Sie bei der Ausgabe durch »printf« die jeweils gleichen Ergebnisse.

```
printf("%d",wert);
printf("%d",*zeiger);
```

Da die Variable »zeiger« nur die Adresse der Variablen »wert« kennt, müssen Sie in der Printf-Funktion wiederum den Operator »*« angeben. Im Fachjargon spricht man hier von dem Objekt eines Zeigers. Ohne diesen Zusatz wird zwangsläufig nur die übertragene Adresse ausgegeben.

In der Programmiersprache C besteht ein enger Zusammenhang zwischen Zeigern und Tabellen. Deutlich wird dies bei der Betrachtung von Zeichenketten. Da kein eigener String-Typ zur Verfügung steht, behilft sich C mit einem Vektor (eindimensionale Tabelle). Um eine Zeichenkette zu vereinbaren, geben Sie an:

```
char string[10];
```

Hierbei repräsentiert das erste Element die Startadresse.

In der Programmiersprache C hängen Zeiger und Tabellen eng zusammen

Dies hat zur Folge, daß die Angabe »&string[0]« bei der Adreßermittlung überflüssig ist. Die alleinige Angabe »string« reicht hierbei aus. Betrachten Sie hierzu nachstehendes Beispiel:

```
char string[10];
char *zeiger_auf_string;
```

Die Anweisung »&string[0]« und die einfache Angabe des Zeigers »zeiger_auf_string« sind vollkommen identisch. In der Vereinbarung beziehungsweise im Speicherbedarf müssen Sie die Unterschiede jedoch beachten. So wird bei Tabellendefinition gleichzeitig Speicherplatz für 10 beziehungsweise 11 Byte (inklusive

C-Funktion	Parameter	Aufgabe
calloc	int Anzahl Elemente int Bytes je Element	reserviert Speicherbereich; besonders für Tabellen geeignet; Vorbelegung 0; Größe = Anzahl x Bytes; diese Funktion liefert die Startadresse des reservierten Bereiches
free	void *Adresse	gibt den durch »malloc« oder »calloc« belegten Speicherplatz frei
malloc	unsigned int Bytes	fordert die angegebene Anzahl von »Bytes« an; als Rückgabe erhält man die Startadresse des reservierten Bereiches
memcpy	void *Adresse_1 void *Adresse_2 unsigned int Bytes	kopiert die angegebene Anzahl an »Bytes« von »Adresse_2« nach »Adresse_1«; diese Funktion liefert die Adresse_1
realloc	void *Speicherbereich unsigned int Byte	ändert die Speichergröße des bisherigen »Speicherbereiches« auf die neue Größe in Bytes ab;

Tabelle. Hilfreiche Funktionen zur Speicherverwaltung

Null-Byte) reserviert. Die Zeigervereinbarung dagegen belegt nur den Platz für die aufzunehmende Adresse – also zwei Byte. Dies hat zur Folge, daß vor der Zuweisung von Werten erst noch Speicherbereich angefordert werden muß.

```
char tab1[3];
char *tab2=" ";
```

Hier wird bei der Zeigervereinbarung gleichzeitig der benötigte Speicherplatz reserviert. Die Variable »tab2« selbst enthält aber dennoch nur die Startadresse der belegten Felder. Nachdem nun alle Unklarheiten beseitigt wären (oder sein sollten), folgt die Zuweisung von einzelnen Zeichen:

```
tab1[0]='D';
*tab2='D';
```

Beide Anweisungen führen zu dem gleichen Ergebnis. Das erste Zeichen innerhalb des Strings enthält nach der Zuweisung nun den alphanumerischen Wert »'D'«. Achten Sie dabei genau auf die Syntax der Zeigerschreibweise. Da keine Adresse übergeben wird, ist in der Zeigervariante der Operator »*« anzugeben. Das Stichwort lautet hierbei: Zeiger auf Objekt.

Die Tabellenform erfordert eine arithmetische, im Zeigerformat reicht eine Inkrement-Anweisung

Sollen nun weitere Zeichen zugewiesen werden, können Sie unterschiedliche Wege einschlagen. Bei der herkömmlichen Verfahrensweise wird nur der Index der Tabelle erhöht. Im Zeigerformat läßt sich diese Form ebenfalls durchführen. Als Alternative bietet sich jedoch eine Inkrementoperation (Erhöhung um 1) an.

```
tab2++;
```

Somit enthält die Zeigervariable »tab2« die Adresse des zweiten Elements.
Wichtig dabei ist, daß bei der Erhöhung immer die Variable »tab2 + +« selbst und nicht das Objekt »*tab2 + +« erhöht wird. Dies würde nämlich ansonsten zu einem sehr seltsamen Resultat führen.

```
tab1[2]='0';
tab2++
*tab2='0';
```

Sie sehen also, der Unterschied ist wirklich minimal. Etwas größer dagegen sind die zeitlichen Differenzen. Während bei der Tabellenform eine arithmetische Anweisung nötig ist (Addition des Index auf die Startadresse), reicht im Zeigerformat eine schnelle Inkrementanweisung aus.
Doch wie wird diese Inkrementlösung intern durchgeführt? Wie Sie wissen, muß mit der Vereinbarung einer Zeigervariablen auch der entsprechende Variablentyp festgelegt werden. Somit erkennt der Compiler, daß eine Ganzzahlvariable vom Typ »int« zwei Byte oder eine Zeichenvariable vom Typ »char« ein Byte belegt. Wenn nun die Adresse einer Char-Variablen um 1 erhöht wird, ist das direkt danebenliegende Byte angesprochen. Wird aber eine Int-Adresse um 1 erhöht, muß zwei Byte weitergesprungen werden. Genauso sieht das Vorgehen bei der Dekrementoperation (Verminderung der Adresse um eine Einheit) aus.
Darüber hinaus können noch weitere Operationen mit Adressen vorgenommen werden. Neben der Addition »tab + 10« ist auch eine Subtraktion »tab-10« möglich. Dies entspricht im Endeffekt dann dem Index einer Tabelle. Doch aufgepaßt – die Grenzen, das heißt Start und Ende eines Datenbereiches, werden nicht abgeprüft. Betrachten Sie einmal folgenden Fall:

```
char *tab="DOS";
char *tab1="International";
tab1-=3;
```

Wird nun das Objekt der Zeigervariablen »tab1« etwa über eine Puts-Funktion ausgegeben, so erhalten Sie folgende Ausgabe:

```
DOS-International
```

Sofern der Compiler die Daten der Reihe nach im Arbeitsspeicher ablegt, führt eine Verminderung der Startadresse »tab1« um drei Byte auf

Utilities gesucht

Programme, die den Bildschirminhalt auf den Kopf stellen, suchen unsere Leser sicherlich nicht.

Aber einen wissenschaftlichen Taschenrechner, ein Festplatten-Backup-Programm, einen Extended-Memory-Treiber, Druckerspooler oder Cache-Speicher.

Zögern Sie deshalb nicht, schicken Sie uns unverbindlich die aktuelle Version Ihres Programms und einen Ausdruck der Dokumentation.

Falls Sie sich vorweg informieren wollen, von

10.00 bis 12.00 Uhr

und

15.00 bis 17.00 Uhr

stehen wir Ihnen an allen Werktagen unter der Rufnummer

(089) 4 61 00 10

zur Verfügung.

Schicken Sie Ihre Unterlagen an folgende Adresse:

Redaktionsbüro
Everts & Hagedorn
»DOS Software«
Hans-Stießberger-Str. 3
8013 Haar/München

das erste Element des Objekts von »tab«.

Nachdem Sie nun wissen, wie man mit den Zeigervariablen umgehen muß, stellt sich wahrscheinlich die Frage nach dem Sinn der Adreß- und Zeigerverwaltung – zumal fast alle Möglichkeiten auch in Form von Tabellendefinitionen vorhanden sind.

Den wesentlichen Vorteil im Umgang mit Zeigern stellt die dynamische Speicherverwaltung dar. Ziel dieser Art der Speicherverwaltung ist es, genau die Menge an Bytes zu reservieren, die im Programm wirklich benötigt wird. Besonders in der Tabellenverarbeitung tritt dieses Phänomen recht häufig auf. Man vereinbart eine Tabelle mit reichlich dimensionierten Feldern, obwohl diese voraussichtlich nie benötigt werden. Folgendes Beispiel macht dies eindrucksvoll deutlich:

```
char tabelle[100][100];
strcpy(tabelle[1],"text");
```

Diese Tabelle benötigt immerhin 10000 Byte Arbeitsspeicherplatz. Gleichzeitig wird in das Element 1 dieser Tabelle die Zeichenkette »text« kopiert, und somit stehen die restlichen 96 beziehungsweise 95 Byte (Null-Byte) unbenutzt, aber reserviert im Speicher. Um dies zu vermeiden, stellt C mehrere Funktionen zur Verfügung. Diese Funktionen sind bei Turbo C in der Include-Datei »stdlib« beziehungsweise »malloc.h« für den Quick-C-Compiler enthalten. Die wesentlichen Routinen finden Sie in der Tabelle.

Wo liegt der Sinn der Adreß- und Zeigerverwaltung – denn als Tabellendefinition geht's auch

Von besonderer Bedeutung ist die Funktion »malloc«. Sie hat die Aufgabe, einen Speicherbereich in der gewünschten Größe (in Byte) zu reservieren. Als maximaler Bereich kann der gesamte Arbeitsspeicher abzüglich

- Betriebssystem (MS-DOS)
- Platz für statische Variablen
- Speicherbedarf der residenten Programme

sowie die Größe des eigenen Programms angegeben werden. Diese Reservierung heißt im Fachjargon auch »allokieren«. Bei Aufruf von »malloc« (memory allocation) versucht die Funktion, den angeforderten Speicher zu reservieren. Falls noch genügend Speicherplatz vorhanden ist, liefert die Funktion einen Zeiger auf die Startadresse des reservierten Bereiches. Ist eine Reservierung nicht möglich, so wird ein Null-Zeiger zurückgegeben. In der Praxis kann dies dann so aussehen:

```
char *zeiger;
char *malloc();
zeiger=malloc(80);
if(zeiger!=NULL)
printf("malloc erfolgreich");
```

Da »malloc« keinen Integer-Wert zurückliefert, muß die Funktion explizit vereinbart werden. Um einen Bereich für Char-Elemente zu reservieren, ist es daher notwendig, die Malloc-Funktion als »Lieferant« eines Zeigers auf »char« (char *malloc();) darzustellen.

Ähnlich arbeitet die Calloc-Funktion. Auch hier wird ein Speicherbereich für eigene Anwendungen reserviert. Zwei kleine – aber feine – Unterschiede sind jedoch zu beachten. Während »calloc« den in Besitz genommenen Speicherbereich mit dem Wert 0 auffüllt, nimmt »malloc« keinerlei Initialisierung vor. Somit liegt eine »Säuberung« ganz in der Verantwortung des Programmierers. Als zweiter Unterschied fällt die Parameterübergabe auf. Bei »malloc« wird eine absolute Größe benötigt, wogegen »calloc« etwas freier agiert.

```
zeiger=calloc(100,sizeof(char));
```

Als erster Parameter wird die Anzahl der Elemente übergeben. Danach folgt die Größe in Byte des einzelnen Elementes. Der hier verwendete Sizeof-Operator ermittelt die jeweilige Größe des Variablentyps. Dies ist in Anbetracht der Portabilität sehr interessant. Da beispielsweise die Größe von Float-Variablen auf den verschiedenen Rechnertypen unterschiedlich ausfällt, wird immer die richtige Anzahl an Byte belegt. Des weiteren müssen Sie sich nicht mit der Berechnung des benötigten Platzes aufhalten.

Für die dynamische Speicherverwaltung stellt C ausgefeilte Funktionen zur Verfügung

Wie eingangs gezeigt wurde, wird bei der Deklaration von Tabellen häufig eine Menge Speicherplatz verschwendet. Als Ersatz für die weiter oben stehende zweidimensionale Tabelle (tab[100][100]) soll nun eine eindimensionale Tabelle vom Typ Zeiger auf »char« angelegt werden. Diese Tabelle kann nun ebenfalls 100 Zeichenketten beziehungsweise deren Startadresse aufnehmen. Zu Beginn des Programms sind aber lediglich 200 Byte (100x2 Byte für Adresse) belegt. Die Länge jeder einzelnen Zeichenkette läßt sich jetzt, je nach Lust und Laune, beliebig variieren.

Betrachten Sie hierzu folgendes Beispiel:

```
char *tab[100];
char *malloc();
tab[1]=malloc(5);
strcpy(tab[1],"text");
```

Die Aufgabe dieser Programmzeilen ist es, dem Tabellenelement 1 die Konstante »text« zuzuweisen. Hierzu sind lediglich fünf Byte inklusive Null-Byte notwendig. Um die ganze Angelegenheit etwas einfacher zu gestalten, ist es möglich, innerhalb der Malloc-Funktion eine andere Funktion aufzurufen.

```
tab[1]=malloc(strlen("text")+1);
```

Mit Hilfe der Funktion »strlen« läßt sich die Länge einer Zeichenkette berechnen. Um auch Platz für das abschließende Null-Byte zu besitzen, wird die Operation » + 1« vor dem Allokieren durchgeführt.

In Verbindung mit der Reservierung von Speicherbereichen ist eine Funktion, die den noch zur Verfügung stehenden Arbeitsspeicher anzeigt, recht sinnvoll. Hierzu stellt Turbo C die Funktion »coreleft« bereit (sie gehört zu Turbo C). Soll beispielsweise ein Bereich reserviert werden, sofern noch mindestens 3000 Byte verfügbar sind, tippen Sie einfach ein:

```
if(coreleft()==3000)
zeiger=malloc(100);
```

In der praktischen Anwendung kommt es häufig vor, daß der allokierte Speicherbereich nicht ausreicht.

Speziell in Textverarbeitungsprogrammen treten derartige Probleme oftmals auf. Aber auch hier hilft die umfangreiche Bibliothek an C-Funktionen. In obigem Bespiel wurde der Zeigervariablen »zeiger« die Startadresse auf 100 Datenfeldern vom Typ »char« zugewiesen. Bedingt durch eine Eingabe des Anwenders ist es aber erforderlich, 200 Byte zu allokieren. Dies realisiert folgende Anweisung:

```
zeiger=realloc(zeiger,200)
```

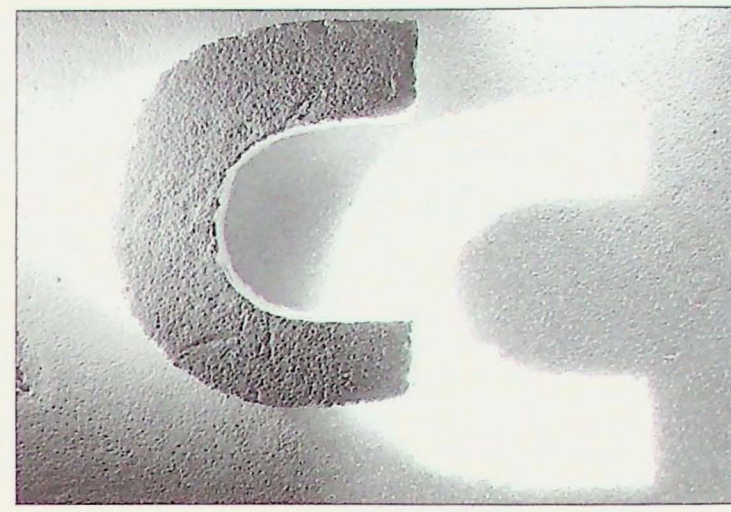
Zum Bedauern jedes Programmierers sind jedem Arbeitsspeicher Grenzen gesetzt. Aus diesem Grunde existiert die Funktion »free«. Als einziger Parameter wird hier die Startadresse des belegten Bereichs übergeben. Soll beispielsweise der oben allokierte Bereich wieder freigegeben werden, so schreiben Sie:

```
free(tab[1]);
```

Vorsicht, die eventuell abgelegten Daten sind mit diesem Aufruf jedoch verloren. Wie Sie sehen, ist der Umgang mit Zeigern und Adressen gar nicht so kompliziert, sofern man die wichtigsten Grundsätze beherzigt.

(Harald Krause/hi)

Flexibel mit Funktionen



Um vernünftige Programme zu schreiben, ist der Einsatz von Funktionen in C unerläßlich. Dieser Beitrag beschäftigt sich daher ausschließlich mit dem sogenannten Modularitätsprinzip.

Die Vorteile der Modularisierung liegen für jeden Programmierer auf der Hand. Hauptsächlich aus Gründen der Übersichtlichkeit ist es ratsam, komplexe Programme in kleine Teile (Module) aufzusplitten. Derartige kleine Module sind aus der Sicht der Fehlerkorrektur (Debugging) wesentlich leichter zu warten.

Ein Beispiel macht die Logik, die sich dahinter verbirgt, vielleicht etwas transparenter. Ein komplexes Programm, beispielsweise ein Dateiverwaltungssystem, besteht aus mehreren Modulen wie Sortierroutinen, Ein- und Ausgabesteuerung und so weiter. Jedes einzelne Modul wird nun separat geschrieben, übersetzt (compiliert) und getestet. Nach erfolgreich beendeter Testphase werden die jeweiligen Teile zusammengebunden (gelinkt), und fertig ist das Programm. Folgt später ein ähnlich gelagertes Problem, können zumindest verschiedene Module wiederverwendet werden.

Die Voraussetzung für eine effektive Modularisierung ist daher, daß einzelne Funktionen zwar vorgegebene Probleme lösen, dies aber relativ allgemein erledigen.

Die Programmiersprache C steht ganz im Zeichen der Modularisierung

Ein wesentlicher Vorteil der Programmiersprache C liegt in der Tatsache, daß der Befehlsvorrat sehr knapp gehalten wurde. Dies erleichert einerseits das Erlernen von C und bietet andererseits ein hohes Maß an Portabilität. Aufgrund der wenigen Befehle ist die Anzahl an nötigen Funktionen zur Ein- und Ausgabe, Konvertierung und Speicherplatzverwaltung um so größer. So stellt beispielsweise Turbo C mehr als 200 und Quick C sogar mehr als 300 Funktionen zur Verfügung. Auf dem Markt der Software-Anbieter sind zudem fertige Routinen für fast alle »Lebenslagen« vorhanden. Eine akzeptable Toolbox für C verschlingt jedoch mit Leichtigkeit eine vierstellige Summe.

Was ist nun eigentlich eine Funktion? Eine Funktion ist eine Sammlung von Befehlen und Anweisungen (meist wieder Funktionen), die mit Hilfe eines beliebigen Funktionsnamens angesprochen werden. Der Funktionsname kann beliebig lang sein, besitzt jedoch nur eine bestimmte Anzahl von signifikanten (wertbaren) Zeichen. Diese Länge hängt vom jeweils vorhandenen Compiler ab. Beträgt die Anzahl der signifikanten Zeichen beispielsweise 16 Zeichen, so sind die beiden nachstehenden Funktionsnamen für den Compiler identisch:

```
berechnung_quadrat1
berechnung_quadrat2
```

Bei der Übersetzung des Programms würden dann entsprechende Fehlerhinweise ausgegeben. Um eine Funktion zu vereinbaren, müssen dem Funktionsnamen zwei runde Klammern »()« folgen. Alle in der Funktion aufgeführten Befehle oder weitere Funktionen werden mit Hilfe der geschweiften Klammern zusammengefaßt. Man spricht hier, genau wie bei Schleifen und Verzweigungen, von einem Block. Das Ende einer Funktion wird entweder durch das Schlüsselwort »return« oder durch das Setzen der schließenden geschweiften Klammer (}) erreicht. Grundsätzlich sollte jede Funktion einen Return-Wert liefern – selbst wenn dies lediglich das Null-Zeichen sein sollte.

```
test_funkt()
{
puts("\n Hallo, Funktion");
}
```

Die Aufgabe der Funktion »test_funkt1« ist es, den Text »Hallo, Funktion« auszugeben. Nach der Ausgabe wird die Funktion verlassen, das heißt, das Programm läuft nach der aufrufenden Stelle weiter. Das folgende Beispiel hilft dabei, die Arbeitsweise von Funktionen zu veranschaulichen:

```
funktion1()
{
puts("\n in Funktion 1");
}

funktion2(){
puts("\n in Funktion 2");
}

main(){
puts("\n main-Funktion");
funktion1();
puts("\n wieder main");
funktion2();
}
```

Dieses kleine Demonstrationsprogramm ruft nacheinander »funktion1« und »funktion2« auf. Jede Funktion gibt hier den aktuellen »Standort« innerhalb des Programms am Bildschirm aus. Als Ausgabe erhalten Sie dementsprechend:

```
in main-Funktion
in Funktion 1
wieder in main
in Funktion 2
```

Während beispielsweise in Pascal zwischen Funktionen (mit Rückgabe) und Prozeduren (ohne Rückgabe) unterschieden wird, findet diese Differenzierung in C nicht statt. Andererseits muß – genau wie bei Variablen – jeder Funktion ein Typ zugeordnet werden. Der Typ einer Funktion entspricht immer dem Typ des Rückgabewerts. Liefert eine Funktion zum Beipiel ein Rechenergebnis vom Typ »double«, so wird die Funktion dementsprechend als »double« vereinbart. Wird kein Typ angegeben, so gilt als Default der Typ »int«. Wie dies in der Praxis aussieht, sehen Sie im nachfolgenden Listing:

```
main(){
double x,y;
double division();
}
```

```
double division()
{
return(-23,98732)
}
```

Der bisher gezeigte Einsatz von Funktionen weist einen gravierenden Nachteil auf. Das Aufgabengebiet ist genau festgelegt, das heißt, jede Funktion läßt sich nur einmal aufrufen. Interessant werden Funktion jedoch erst dann, wenn die Steuerung variabel ist. Dazu ist es notwendig, Argumente beziehungsweise Parameter an eine Routine zu übergeben und die Ergebnisse (Return) auszuwerten. Wie Sie bereits anhand der Puts-Funktion gesehen haben, wird der Übergabewert (hier eine Zeichenkette) innerhalb der runden Klammern eingetragen. Bei der Funktionsvereinbarung müssen zusätzlich die Typen der zu erwartenden Parameter definiert sein. Im Falle der Puts-Funktion handelt es sich um einen Zeiger auf »char«.

```
addition(para1, para2)
int para1, para2;
{
printf("%d",para1+para2);
}
```

Diese Funktion erwartet also zwei ganzzahlige Werte vom Typ »int«. Ein Aufruf dieser Funktion aus dem Hauptprogramm kann etwa so lauten:

```
main(){
addition(3,4);
}
```

Somit werden an die Funktion zwei Werte, nämlich 3 und 4, übergeben. Innerhalb dieser Funktion nimmt nun die Variable »para1« den Inhalt 3 und die Variable »para2« den Inhalt 4 an. Die Variablennamen der Parameter können Sie freilich beliebig wählen. Selbst Namensgleichheit zwischen aufrufender und gerufener Funktion ist möglich. Wird dagegen ein anderer Typ wie etwa »char« oder »double« verwendet, tritt zum Übersetzungszeitpunkt ein Fehler auf. Natürlich können auch Variablen beziehungsweise deren Adressen übergeben werden.

```
main(){
int a=3,b=4;
addition(a,b);
}
```

Dieses Programm arbeitet genau wie das vorherige. Sinn einer Funktion ist aber nicht nur die Entgegennahme von Parametern, sondern auch die Rückgabe. Denn was nützt eine Funktion, die Werte entgegennimmt, ein Ergebnis berechnet, es aber nicht zurückgibt. Um dies zu bewerkstelligen, steht der Befehl »return« zur Verfügung. Die allgemeine Form hierzu lautet:

```
return (Rückgabe)
```

Manche C-Compiler gestatten es auch, die runden Klammern wegzulassen; diesbezüglich sehen Sie am besten im Compiler-Handbuch nach. Die Anwendung des Return-Befehles erlaubt nur die Rückgabe eines Wertes beziehungsweise einer Adresse. Zur Verdeutlichung dient folgendes Beispiel:

```
int addition(para1,para2)
int para1,para2;{
return(para1+para2);
}
```

Die aufrufende Main-Funktion hat nun zwei Möglichkeiten, das Ergebnis auszuwerten. Das Resultat kann in einer Variablen abgelegt werden oder direkt an eine andere Funktion weitergegeben werden.

```
main(){
int a=3,b=4,c;
c=addition(a,b);
printf("%d %d",c,addition(a,b));
}
```

Beide Varianten sind somit gleichwertig. Übrigens läßt sich auch der Funktionsaufruf verschachteln. Dies ist in der C-Programmierung leider gängige Praxis. Theoretisch kann mit einem einzigen verschachtelten Funktionsaufruf ein ganzes Programm geschrieben werden. Doch sehen Sie selbst:

```
printf("%f%",pow(atoi(gets(buf))));
```

Aufgabe dieses mustergültigen Programms ist es, einen Char-Wert von der Tastatur zu holen, diesen zu einem ganzzahligen (Integer-) Wert umzuwandeln, anschließend das Quadrat zu bilden und auf dem Bildschirm auszugeben. Diese Reihe ließe sich natürlich noch beliebig fortsetzen.

Der Chaos-Programmierung ist Tür und Tor geöffnet – Disziplin ist gefragt

Doch nun weiter im Text. Genau wie in der Main-Funktion, dürfen Variablen auch in »normalen« Funktionen vereinbart werden. Man spricht hier von automatischen oder lokalen Variablen. Vorsicht ist beim Verlassen der Funktion geboten, denn sämtliche Variableninhalte sind nun verloren. Eine Methode, den Datenverlust zu verhindern, besteht im Ändern der Speicherklasse. Es bieten sich hierzu drei verschiedene Speicherklassen an:

- auto
- static
- extern

Bei der Deklaration einer Variablen ohne explizite Angabe der Speicherklasse wird generell die Auto-Klasse gewählt. Die Vereinbarung von »auto int a = 4« ist somit identisch mit »int a = 4«.

Häufig ist es aber wünschenswert, daß die Variableninhalte auch nach dem Verlassen einer Funktion erhalten bleiben. Hierzu bietet sich die Speicherklasse »static« an. Derartig deklarierte Variablen werden in einem besonderen Arbeitsspeicherbereich abgelegt. Zudem findet mit der Deklaration auch eine Vorbelegung statt. Arithmetische Typen (int, float und double) werden auf 0 gesetzt und Zeichenketten mit Leerzeichen aufgefüllt.

Eine weitere Problematik bezüglich Funktionen stellt der Geltungsbereich von Variablen dar. Auf Variablen, die innerhalb einer Funktion vereinbart wurden, kann weder die Hauptfunktion (main) noch eine sonstige Funktion zugreifen. Auch Namensgleichheit beeinflußt den »lokalen Geltungsbereich« nicht. Zwar können mehrere Variablen als Parameter übergeben werden – zurück kommt jedoch nur eine.

Aus diesem Grunde bietet C die Möglichkeit der »globalen Variablen«. Eine globale Variable wird außerhalb von Funktionen deklariert und von jeder sich in einem Modul befindlichen Funktion erreicht.

```
int prod;

main(){
int i;
extern prod;
prod=3;
printf("%d",add(3,4));
printf("\t %d",prod);
}

int add(a,b)
int a,b;{
extern summe;
prod*=a;
return(a+b)
}
```

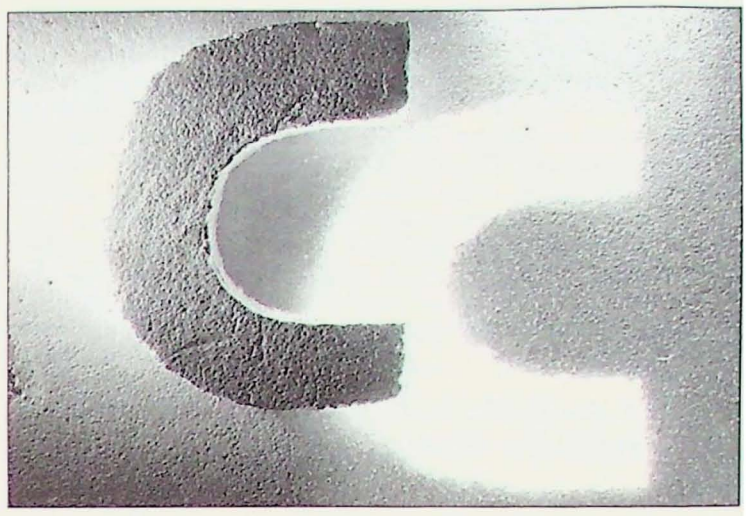
Damit die Funktion derartige globale Variablen erkennt, sollten Sie explizit das Schlüsselwort »extern« verwenden. Dies ist zwar nicht unbedingt erforderlich, doch die Lesbarkeit erhöht sich dadurch wesentlich. Betrachten Sie hierzu obiges Beispiel.

Die Variable »prod« wird außerhalb der beiden Funktionen deklariert und kann deshalb von »main« und »add« erreicht werden. Der Geltungsbereich von »prod« ist somit global. Während in »main« der Variablen »prod« der Wert 3 zugewiesen wird, multipliziert die Add-Funktion diesen Wert mit der Variablen »a« (3). Als Ausgabe der Printf-Funktion erhalten Sie nun die Zahlen 7 und 9.

Versuchen Sie aber niemals, gleiche Namen für lokale und globale Variablen zu verwenden, denn dies führt garantiert zu großen Problemen. Hüten Sie sich auch vor zu häufigem Einsatz von globalen Variablen, da die Übersichtlichkeit Ihres Programms beträchtlich darunter leidet.

(Harald Krause/hi)

Präprozessor – der kleine Helfer



Jedes C-System besteht aus zwei Übersetzern – dem Compiler und dem Präprozessor. Die Arbeitsweise eines Compilers ist den meisten bekannt – die Funktion des Präprozessors gibt hingegen häufig noch Rätsel auf.

Zum Lieferumfang eines C-Pakets gehören die verschiedensten Tools. Die wichtigsten sind der Compiler, der Linker sowie verschiedene Library-Hilfsprogramme. Ein relativ unbekanntes Instrumentarium stellt der in jedem C-Paket enthaltene Präprozessor dar. Hierbei handelt es sich um ein leistungsfähiges Hilfsmittel für den engagierten Programmierer.

Der Präprozessor steht mit mehr oder minder komfortablen Funktionen zur Verfügung. Leider werden diese Funktionen nicht in allen Systemen geboten. Um im Rahmen dieses Beitrags die wesentlichen Eigenschaften anzusprechen, soll nur auf allgemeingültige Anweisungen eingegangen werden.

Was ist eigentlich ein Präprozessor? Ein Präprozessor, deutsch »Vorübersetzer«, arbeitet unabhängig vom eigentlichen Compiler. Bevor ein Programm vom Compiler übersetzt wird, tritt der Präprozessor in Erscheinung. Jedes Präprozessor-Statement wird durch das Doppelkreuz (#) eingeleitet. Hierbei gibt es verschiedene Möglichkeiten. Eine der meistbenutzten Funktionen haben Sie sicherlich schon einmal angewendet – die Include-Anweisung. Wie Sie wissen, kopiert
dieser Befehl eine angegebene Datei in ein C-Programm.

```
#include <stdio.h>
```

Dieser Befehl bewirkt, daß die Bibliotheksdatei »stdio.h« im aktuellen Verzeichnis gesucht und anschließend (falls vorhanden) eingefügt wird. Soll die Suche dagegen in einem anderen Verzeichnis erfolgen, so muß dieser Pfad »absolut« angegeben werden.

In der Syntax sind hierbei einige Änderungen zu beachten. Anstatt des Kleiner- (<) und Größer-Zeichens (>) werden Anführungszeichen (") verwendet. Eine korrekte Schreibweise könnte etwa folgendermaßen lauten:

```
#include "\c\biblio.c"
```

Es ist immer notwendig, den kompletten Pfad anzugeben.

```
#include "..\biblio.c"
```

Obiges Beispiel führt mit Sicherheit nicht zum gewünschten Ergebnis.

Die Alternative zu Variablen wird mit dem »define« geboten

Eine weitere Anwendungsmöglichkeit stellt die Define-Anweisung dar. Der Sinn dieses Befehls liegt darin, Konstanten zu Beginn oder im Laufe des Textes zu vereinbaren. Diese Konstanten werden während der Vorübersetzung durch den dahinter angegebenen Wert (oder die Zeichenfolge) ersetzt. Speziell für Umsteiger von Pascal auf C ergibt sich hieraus eine gewisse Erleichterung. Während in Pascal Anfang und Ende jedes Anweisungsblocks oder jeder Funktion durch die Schlüsselworte »begin« und »end« markiert werden, läßt sich dies mit Hilfe der Define-Anweisung ebenfalls erreichen. Schreiben Sie einfach

```
#define BEGIN {
#define END }
```

und es besteht hinsichtlich der Blockstrukturen kein Unterschied mehr. Im Programmtext sieht dies nun folgendermaßen aus:

```
main()
BEGIN
puts("DOS-Extra");
END;
```

Nach der Vorübersetzung erhält der C-Compiler den korrekten C-Code.

```
main
{
puts("DOS-Extra");
}
```

Aber nicht nur für Pascal-Umsteiger ist der Define-Befehl von Nutzen. Auch professionelle C-Programmierer arbeiten in jedem Programm mit dieser Hilfe. Warum sie das tun, zeigt ein kleines Beispiel:

```
#define MAX 100
main(){
int tabelle[MAX];
int i;
for(i=1;i>MAX;i++)
...
```

Der Define-Befehl weist der »Konstanten« den Wert 100 zu. Während des Präprozessorlaufs werden alle im Text befindlichen MAX-Angaben durch diesen Wert ersetzt. Nach dem Ersetzen sieht der Quellcode dann folgendermaßen aus:

```
main(){
int tabelle[100];
int i;
for(i=1;i<100;i++)...
```

Befehl	Aufgabe	Anwendung
define	– reiner Textersatz – Makroprogrammierung	#define TEXT1 TEXT2 #define MAK(x) Operation
undef	hebt Gültigkeit von #define wieder auf	#define TEXT1 TEXT2 #undef TEXT1
include	kopiert bestehende Datei in das aktuelle Programm	#include <stdio.h> #include "\stdio.h"
if	führt Befehle nach Bedingung aus, sofern Wert ungleich ist	#define WERT 1 #if WERT puts("ungleich 0"); #endif
endif	beendet bedingte Compilierung, die durch #if eingeleitet wurde	siehe oben
ifdef	leitet bedingte Compilierung ein, bestimmte #define Konstante vorhanden ist;	#define JA #ifdef JA puts("JA vorhanden"); #else (oder #endif); puts("JA nicht vorhanden"); Ende: #endif oder #else #endif
ifndef	leitet bedingte Compilierung ein, bestimmte #define Konstante nicht vorhanden ist;	#define JA #ifndef JA puts("JA nicht vorhanden"); #else (oder #endif); puts("JA vorhanden"); Ende: #endif oder #else #endif

Tabelle. Interessante Befehle für den Präprozessor

Hierbei wird klar, daß die Bezeichnung »Konstante« eigentlich nicht korrekt ist; es wird nämlich kein Speicherplatz belegt.
In der Praxis findet der Define-Befehl rege Verwendung. Die Vorteile liegen klar auf der Hand. Es ist wesentlich einfacher, eine solche Vereinbarung nur einmal im Text zu verändern (» # define MAX 200«), als den gesamten Programmtext zu durchsuchen, denn bestimmt vergißt man zumindest eine Stelle.

In C ist auch die Vereinbarung größerer Makros erlaubt

Neben diesem reinen Ersetzverfahren ist es auch möglich, größere Makros zu vereinbaren. Ein Makro ist eine Ansammlung von verschiedenen Befehlen und entspricht somit in etwa einer Funktion. Diese Zusammenfassung wird analog zu Funktionen mit einem einzigen Namen angesprochen. Lediglich im Speicherbedarf bestehen Unterschiede. Während eine Funktion innerhalb eines Programms nur ein einziges Mal vorkommt, wird ein Makro immer an die aufrufende Stelle kopiert. Benötigen Sie ein Makro sehr häufig, so empfiehlt es sich hinsichtlich der Programmgröße (und damit Geschwindigkeit), eine eigenständige Funktion zu entwickeln.
Nehmen Sie an, in einem Programm ist es nötig, mehrmals das Quadrat einer Integer-Zahl zu bilden. Weil dies eine recht einfache Aufgabenstellung ist, soll auf eine eigene Funktion verzichtet werden. Nachstehendes Makro löst das Problem auf ganz simple Art und Weise:

```
# define QUADRAT(x) (x*x)
```

Das Makro wird mit dem Namen »QUADRAT« angesprochen, und der Variablen x wird ein Wert übergeben. Die eigentliche Rechenanweisung wird durch ein Leerzeichen oder einen Tabulatorsprung in Klammern angegeben. Ein Aufruf dieses Makros geschieht folgendermaßen:

```
main(){
printf("%d",QUADRAT(3));
}
```

Da der Präprozessor diese Berechnung nicht selbst durchführen kann, erfolgt ein Textersatz in der Printf-Funktion. Der Compiler findet nun folgenden Text vor:

```
printf("d",(3*3));
```

Selbstverständlich können Sie auch mehrere »Variablen« angeben. So gibt es keine Probleme, auch komplexe Rechenaufgaben oder Funktionsaufrufe per Makro zu vereinbaren.

```
# define TEST(x,y)
printf("%d %d",x,y)
```

Diese Anweisungen haben zur Folge, daß zwei Werte (x und y) formatiert ausgegeben werden.

Es macht keine Probleme, komplexe Rechenaufgaben per Makro zu erledigen

Im Gegensatz zum C-Compiler wird beim Präprozessor sehr stark auf die Syntax geachtet. Schon ein einziges Leerzeichen führt manchmal zum Fehler. So darf zum Beispiel zwischen dem Doppelkreuz und dem Befehl niemals ein Leerzeichen stehen. Genauere Informationen zur Präprozessor-Syntax finden Sie im jeweiligen Compiler-Handbuch.
Nach dem ungeschriebenen Gesetz von Kerningham und Ritchie werden alle Define-Konstanten groß geschrieben. Im Interesse der Einheitlichkeit sollten auch Sie sich an diese Regelung halten.

(Harald Krause/hi)

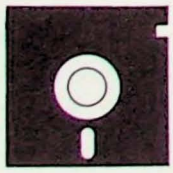
DOS INTERNATIONAL

Zur Ergänzung unseres jungen, engagierten Redaktionsteams suchen wir einen versierten

Fachredakteur/in

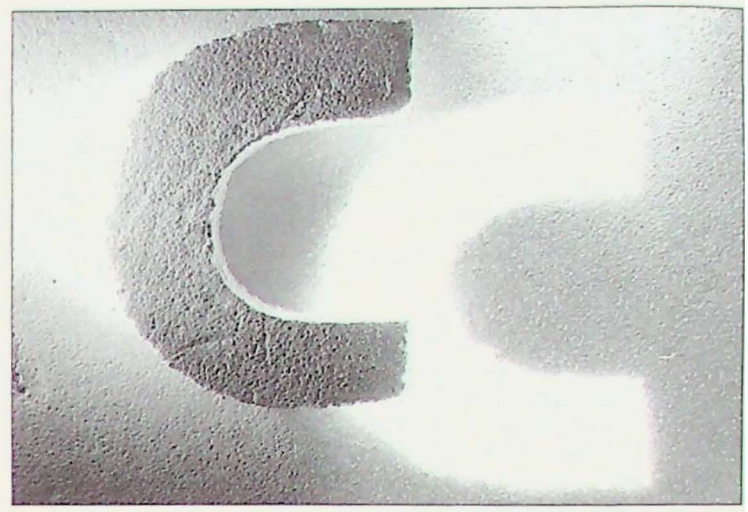
Sie haben bereits grundlegende Erfahrungen im journalistischen Bereich, fundierte PC- und Softwarekenntnisse, Sie drücken sich gewandt aus und bringen die Bereitschaft mit, selbständig zu arbeiten.

Sind Sie interessiert? Dann senden Sie uns Ihre Bewerbungsunterlagen. Wir freuen uns auf ein persönliches Gespräch.



Redaktionsbüro Everts & Hagedorn
Redaktion DOS International
Hans-Stießberger-Str. 3
8013 Haar/München

Vektoren und Tabellen in C



Ein zentrales Thema in der C-Programmierung stellt die Tabellenverarbeitung dar. Im Gegensatz zu anderen Programmiersprachen wird eine Zeichenkette (String) in C als Tabelle beziehungsweise als Vektor abgebildet.

Beinahe alle Programme nutzen die Möglichkeiten, die sich durch den Einsatz von Tabellen eröffnen. Prinzipiell entspricht eine Tabelle einer Reihe von Datenfeldern, die mit Hilfe eines Index angesprochen werden. Einzige Voraussetzung hierfür ist, daß jedes Feld denselben Variablentyp besitzt. Das heißt, wenn eine Tabelle als »Integer« vereinbart wurde, führt eine Zuweisung von Char-Werten folglich zu einem sinnlosen Ergebnis.
Um eine Tabelle in C zu deklarieren, muß nach Vergabe des Variablennamens ein eckiges Klammerpaar angehängt werden. Innerhalb dieser Klammern steht die maximale Ausprägung dieser Tabelle. Hierbei muß es sich um einen ganzzahligen positiven Wert handeln. Betrachten Sie folgende Bespiele:

```
int tabelle_1[10];
float tabelle_2[10];
```

»tabelle__1« entpricht also einer Reihe von zehn ganzzahligen Datenfeldern und »tabelle__2« einer Reihe von zehn Fließkommafeldern. Der Typ einer Tabelle kann genau wie jede Variable (also ein Datenfeld) deklariert werden. So besteht auch die Möglichkeit, eine Tabelle aus Zeigern auf »char« (char *tab[10]) oder jede andere beliebige Konstellation zu vereinbaren.
In C beginnt eine Tabelle immer mit dem Index 0. Daraus folgt, daß das letzte Element genau dem in der Klammer angegebenen Wert abzüglich 1 entspricht. Soll also das letzte Element der »tabelle__1« angesprochen werden, so lautet der Befehl zum Beipiel:

```
tabelle_1[9]=100;
```

Hier wird nun dem zehnten Element der Wert 100 zugewiesen.
Das Bearbeiten oder Anzeigen von Tabellenwerten entspricht dagegen wieder »normalen« Konventionen. Um den Inhalt eines Tabellenelementes am Bildschirm anzuzeigen, können Sie folgende Anweisung verwenden:

```
printf("9: %d",tabelle_1[9]);
```

Das folgende Listing verdeutlicht die Arbeitsweise von Tabellen. Sinn und Zweck ist es, die Summe sowie den Durchschnitt einer Wertetabelle zu berechnen.
Die maximal 50 Zahlen werden mit Hilfe der Funktion »scanf« eingelesen. Als Endekriterium dient entweder die eingegebene Ziffer -1 oder das Erreichen des Tabellenendes.

Selbst Zeichenketten werden in C als Tabellen angesehen

Da sich jede Tabelle aus einer Reihe von Datenfeldern zusammensetzt, geschieht die Vorbelegung analog zu einfachen Variablen. Das heißt, der Inhalt einer Tabelle ist bei Programmstart völlig ungewiß. Abhilfe schafft hier eine Vorbelegung, die durch zwei Varianten erreicht werden kann:
1. Eine Initialisierung nach eigenen Vorstellungen. Nachstehendes Beispiel zeigt dies.
2. Mit Hilfe der Speicherklasse »static« findet grundsätzlich eine Vorbelegung auf 0 statt.

```
int
tabelle_1[10]={1,2,3,4,5,6,7,8,9,0};
```

Die Angabe der Tabellengröße ist hierbei optional. Sie könnten also ebenso schreiben:

```
int tabelle_1[]={1,2,3,4,5,6,7,8,9,0};
```

Beide Schreibweisen sind hierbei völlig identisch, da der Compiler die Größe der Tabelle anhand der Anzahl der Elemente bestimmen kann. Dies bedeutet aber auch, daß bei einer teilweisen Vorbelegung die Tabelle zu klein angelegt wird. Wird die Tabelle bereits bei der Definition als »static« erklärt, so ist zumindest gewährleistet, daß alle einzelnen Elemente den Wert Null besitzen.
Die Sprache C betrachtet eine Zeichenkette immer als Vektor (eindimensionale Tabelle).

```
char string[3]
```

Die obige Schreibweise steht für die Vereinbarung einer Zeichenkette, die drei Zeichen aufnehmen kann. Analog zur Verfahrensweise bei arithmetischen Tabellen entspricht das erste Zeichen dem Index 0. Die Zuweisung ist ebenfalls identisch.

```
string[0]='D';
string[1]='O';
string[2]='S';
string[3]='\0';
```

Ein derartig aufgebauter String läßt sich nun mit der Puts- oder Printf-Funktion ausgeben. Tippen Sie den Befehl

```
puts(string);
```

ein, so erscheint am Bildschirm die Ausgabe »DOS«. Im Gegensatz zu »normalen« Tabellen ist das Ende eines Strings gekennzeichnet, und zwar mit dem Null-Byte (»\0«).
Wird beispielsweise durch eine fehlerhafte Zuweisung das Null-Byte überschrieben, so erhalten Sie in den meisten Fällen eine chaotische Ausgabe. Probieren Sie es einmal aus – fügen Sie einfach »string[3] = 'i'« in Beispielanweisungen ein, und geben Sie sie ebenfalls über die Puts-Funktion aus.
Um die Arbeitsweise eines Compilers zu verstehen, ist es wichtig, die einzelnen Abläufe zu kennen. Bei der Ausgabe eines Strings führt der Compiler folgende Schritte durch:

- Er holt die Adresse der Zeichenkette (Element Null).
- Ist das aktuelle Zeichen ungleich dem Null-Byte?
- Wenn ja, Ausgabe des Zeichens.
- Wenn nein, Ende der Ausgabe.

Diese Reihe von Prüfungen und Befehlen können Sie auf einfache Weise selbst nachvollziehen. Die hierzu benötigte Funktion »putchar« hat die Aufgabe, ein einzelnes Zeichen am Bildschirm auszugeben.
Um sich die Zuweisung der einzelnen Zeichen zu ersparen, ist es ratsam, die Funktion »strcpy« (string copy) anzuwenden. Diese Funktion hängt nämlich beim Kopiervorgang automatisch das Null-Byte an. Das

nachfolgende Programm simuliert die Puts-Funktion:

```
#include <stdio.h>

main(){
int i=0;
char string[3];
strcpy(string,"DOS");
while(string[i]!=NULL){
putchar(string[i]);
i++;
}
}
```

Besondere Probleme ergeben sich bei der Verarbeitung von Strings. Bedauerlicherweise ist es nicht möglich, eine Zeichenkette innerhalb einer Funktion zuzuweisen. Folgende Anweisung ist daher nicht möglich:

```
main(){
char string[3]="DOS";
}
```

Als typische Fehlermeldung erscheint in diesem Fall: »cannot initialize array in function«. Grund dafür ist erneut die Adreß- beziehungsweise Zeigerverwaltung. Das Element Null (string[0]) stellt nämlich die Startadresse einer Zeichenkette dar. Eine Lösungsmöglichkeit ist daher, einen String außerhalb von Funktionen zu initialisieren.

```
char string[3]="DOS"
main(){
...
```

Neben den bisher angesprochenen eindimensionalen Tabellen kennt C selbstverständlich auch mehrdimensionale Tabellen. Im Regelfall unterstützt ein C-Compiler bis zu 15 Dimensionen, wie auch beispielsweise Quick C. Die maximale Tabellengröße ist in den normalen Speichermodellen (»small«, »medium« und »compact«) auf ein Segment (64 KByte) beschränkt. Die Syntax der mehrdimensionalen Tabellenvereinbarung lautet folgendermaßen:

```
tabellentyp name
[dim1][dim2][dim3][..];
```

In der praktischen Anwendung wird eine zweidimensionale Tabelle vom Typ »int« so deklariert:

```
int tabelle_3 [3][5];
```

Diese Tabelle besteht somit insgesamt aus 15 Datenfeldern, die einen Speicherplatz von 30 Byte beanspruchen. Jedes einzelne Element muß mit Hilfe von zwei Indizes angesprochen werden. Das niedrigste Datenfeld wird mit »tabelle__3 [0][0]« und das letzte beziehungsweise höchste mit »tabelle__3 [2][4]« erreicht. Das folgende kleine Programm zeigt, wie sich mehrdimensionale Tabellen einsetzen lassen:

```
main(){
int tab [3][5];
int i1,i2;
for(i1=0;i1<3;i1++)
for(i2=0;i2<5;i2++)
tab [i1][i2]=i1+i2;
}
```

Analog zu eindimensionalen Datenfeldern besteht auch hier das Problem der nichtvorhandenen Vorbelegung. Dies läßt sich jedoch ähnlich lösen. Entweder Sie deklarieren die Tabelle als »static«, oder Sie behelfen sich mit der manuellen Initalisierung. Bedauerlicherweise bestehen hier zwischen den einzelnen Compilern diverse Unterschiede. Folgender Programmteil führt in den meisten Fällen zum Erfolg:

```
int tab[3][5]={
{1,2,3,4,5},
{2,3,4,5,6},
{3,4,5,6,7}
};
```

Jede einzelne Tabellenreihe (Hauptindex) wird mit den geschweiften Klammern zusammengefaßt. Manche Compiler, wie Turbo C oder Quick C, verzichten auf derartige Gliederungen.

Weil C das Modularitätsprinzip voll unterstützt, ist die Übergabe von Tabellen beziehungsweise deren Inhalt an Funktionen notwendig. Diese Übergabe gestaltet sich sehr einfach:

```
main(){
int tabelle[5];
funktion(tabelle);
{

funktion(tab)
int tabelle[];
{
...
}
```

Die Angabe der Tabellengröße können Sie bei der aufgerufenen Funktion weglassen. Wird jedoch innerhalb der aufgerufenen Funktion eine Größenangabe vorgenommen, so muß diese identisch mit der Deklaration sein.

Tabellen in C bergen noch ein weiteres Problem. Es ist ohne weiteres möglich, die Grenzen einer Tabelle zu überschreiten. Nehmen Sie hierzu wieder obige Datenfelddefinitionen. Für die »tabelle__1« stehen lediglich zehn einzelne Felder zur Verfügung. Wird aber, etwa durch einen Tippfehler oder eine fehlerhafte Schleifenkonstruktion, das Element 10 (tabelle__1[10]) angesprochen, so »merkt« dies weder der Compiler noch der Linker.

Stellen Sie sich den Arbeitsspeicher in der Gesamtheit vor, so erscheint dieser lediglich als eine Reihe einzelner Bits beziehungsweise Bytes. Wird nun eine Tabelle vereinbart, reserviert das Betriebssystem die gewünschte Anzahl von Bytes. Die Startadresse dieser Tabelle übermittelt das Betriebssystem nun dem Programm. Die Grenze (Tabellenende) ist dagegen nirgendwo festgelegt. Wird nun beispielsweise dem Element 1 ein Wert zugewiesen, so sucht das Programm nur die Startadresse der Tabelle (tabelle__1[0]) und addiert dazu den Index mal die Länge des einzelnen Feldes (»int« meist zwei Byte). Dabei ist völlig gleichgültig, auf welche tatsächliche Speicherstelle zugegriffen wird.

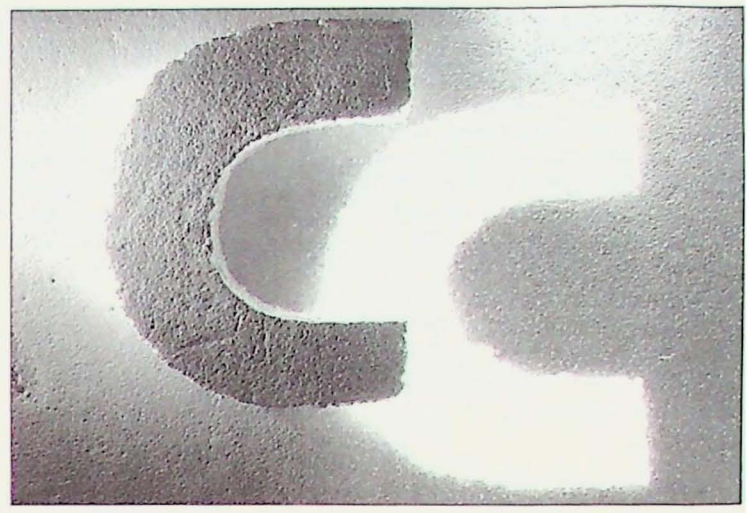
(Harald Krause/hi)

```
[TP6]    1: int werte_tab[3];    /* Globale Tabelle */
[ØØØ]    2:
[5C2]    3: int einlesen()
[P7Ø]    4: {
[Ø14]    5:  int i;    /* Laufvariable */
[ØØØ]    6:
[9W2]    7:  for(i=1;i<=3;i++){
[9A4]    8:    printf("Wert Nr: %d \t",i);
[5E4]    9:    scanf("%d",&werte_tab[i-1]);
[6M4]   1Ø:     /* Tabelle beginnt bei Index Ø */
[NK3]   11:    if(werte_tab[i-1]==-1)
[FK2]   12:     /* Abbruch bei -1 */
[VU1]   13:      return(i-1);
[134]   14:       /* letzte Eingabe bleibt */
[WL3]   15:       /* unberücksichtigt */
[U7Ø]   16:  }
[KM1]   17:  return(i);
[U7Ø]   18: }
[ØØØ]   19:
[ØØØ]   2Ø:
[545]   21: int rech_und_ausgabe(zaehler)
[952]   22:    int zaehler;
[P7Ø]   23: {
[RN2]   24:   double summe=Ø.Ø;
[ØH3]   25:   double durchschnitt;
```

```
[F11]   26:   int i;
[8L3]   27:   for(i=Ø;i<zaehler;i++)
[HP3]   28:     summe+=werte_tab[i];
[A83]   29:      /* Summe ermitteln */
[ØØØ]   3Ø:
[6Y4]   31:   durchschnitt=summe/zaehler;
[1B8]   32:   printf("\nEs wurden %d Werte eingegeben",
          zaehler);
[RL8]   33:   printf("\nSumme: %f Mittel: %f",summe,dur
          chschnitt);
[U7Ø]   34: }
[ØØØ]   35:
[ØØØ]   36:
[UWØ]   37: main()
[P7Ø]   38: {
[952]   39:  int zaehler;
[4A3]   4Ø:  zaehler=einlesen();
[UY4]   41:   /* Anzahl der gelesenen Werte */
[AK4]   42:  rech_und_ausgabe(zaehler);
[U7Ø]   43: }
```

Listing. Berechnung von Summe und Durchschnitt einer Wertetabelle

Dateizugriff in C



Hauptsächlich wird C in Verbindung mit technischen und mathematischen Problemen erwähnt. Daß nicht allein darin die Stärke dieser Sprache besteht, zeigt die Tatsache, daß selbst das relationale Datenbanksystem dBase in C geschrieben wurde.

Bevor Sie darangehen, sich »Ihr« dBase mit C selbst zu programmieren, benötigen Sie Informationen über einige Feinheiten des Datei-Handlings. Eine Besonderheit der Sprache C liegt darin, daß jedes externe Gerät als Datei betrachtet wird. So stellt selbst die Tastatur eine »Datei« dar. Wie erwähnt, bezeichnet man die Tastatur nicht als solche, sondern verwendet den Begriff der Standardeingabe. Gleiches gilt für die Standardausgabe – dies ist in der Regel der Bildschirm. In der Standardbibliothek (»stdio.h«) sind verschiedene Standardkanäle bereits definiert. So kennt man

- Standardausgabe = »stdout«
- Standardeingabe = »stdin«
- Standardfehler = »stderr«
- Drucker = »stdprn«
- Serieller Port = »stdaux«

Hinter diesen Bezeichnungen steht, ebenso wie im Betriebssystem, eine Nummer. Im Fachjargon spricht man hierbei von »Handles«. Wenn Sie in der Datei »stdio.h« Ihres Compilers nachsehen, so finden Sie dort die Nummern der Standardgeräte. Um den Handle einer Datei zu erhalten, ist es erforderlich, diese Datei zu öffnen. (Bei einem Standardgerät ist dies natürlich überflüssig).

```
handle=open("xyz.dat");
```

Diese Dateinummer wird immer dann benutzt, wenn Ein- oder Ausgaben durchgeführt werden sollen. Die Anzahl der möglichen Dateien ist abhängig vom Eintrag in der Initalisierungsdatei »config.sys«

```
files=10
```

Hierdurch erkennt MS-DOS, daß maximal zehn Dateien eröffnet werden können. Von diesen zehn Dateien werden nun die oben aufgelisteten Hardware-Dateien abgerechnet. Mit dieser Numerierungstechnik lassen sich einige verblüffende Ergebnisse erzielen. Betrachten Sie hierzu folgendes Beispiel:

```
fprintf(stdout,"Hallo C");
```

Die Funktion arbeitet genauso wie die bereits bekannte Printf-Funktion; es wird also ein formatierter String ausgegeben. Lediglich der Ausgabeort kann verlegt werden. Mittels »stdout« wird der Text »Hallo C« an den Standardausgabekanal geleitet. Setzen Sie hier nun etwa den Namen einer Datei (beziehungsweise Handle) oder die Bezeichnung für den Druckerport »stdprn« ein, so wird die Ausgabe einfach umgeleitet.

In C sind alle Dateiverwaltungsfunktionen in zwei Kategorien aufgeteilt. Zum einen werden die Basis- oder Low-Level-Routinen und zum anderen die High-Level-Routinen oder gehobenen Funktionen benutzt. Da jedoch die Low-Level-Routinen sehr hardwarenah arbeiten, sollen diese Funktionen im Rahmen dieses Beitrags nicht weiter besprochen werden.

Im Gegensatz zu anderen Programmiersprachen sieht C jedes Gerät als Datei an

Alle High-Level-Funktionen erfordern immer das Einbinden der Datei »stdio.h«. In dieser Include-Datei befindet sich nämlich eine spezielle Typvereinbarung namens »FILE«. Dieser besondere Typ nimmt den sogenannten »Filepointer« auf. Dieser Filepointer ist eine erweiterte Variante des oben besprochenen Handles.

Während in anderen Programmiersprachen, wie beispielsweise Cobol, umfangreiche Befehle zum Datei-Handling zur Verfügung stehen, müssen Sie dies in C selbst organisieren. Lediglich verschiedene Grundfunktionen der Dateiverwaltung sind hier vorhanden. So können Sie zwar problemlos Dateien anlegen, beschreiben oder lesen – die einzelnen Datensätzen jedoch müssen Sie selbst wiederfinden.

r	Öffnen zum Lesen
w	Öffnen zum Schreiben
a	Öffnen zum Erweitern
r+	Öffnen zum Schreiben und Lesen, wobei die zu eröffnende Datei bereits bestehen muß
w+	Öffnen zum Schreiben und Lesen; eine bereits bestehende Datei wird überschrieben
a+	Öffnen zum Erweitern; besteht die Datei noch nicht, so wird sie angelegt

Tabelle 1. Alle denkbaren Eröffnungsmodi im Rahmen der High-Level-Funktionen

O_APPEND	positioniert den Dateizeiger auf das Ende der Datei, bevor geschrieben werden kann
O_BINARY	öffnet Datei im Binärmodus; es werden keine Konvertierungen vorgenommen
O_CREAT	legt eine neue Datei an und öffnet diese; falls die Datei bereits existiert, wird keine Öffnung vorgenommen
O_RDONLY	eröffnet Datei nur zum Lesen
O_RDWR	eröffnet Datei sowohl zum Lesen als auch zum Schreiben
O_TEXT	öffnet Datei im Textmodus; hierbei wird die Kombination CR-LF (carriage-return-line-feed) in LF (line-feed) umgewandelt
O_TRUNC	eröffnet eine Datei und setzt die Länge anschließend auf Null
O_WRONLY	eröffnet Datei zum Schreiben

Tabelle 2. Alle denkbaren Eröffnungsmodi im Rahmen der Low-Level-Funktionen

Diese Tatsache bedeutet aber zugleich, daß Sie seitens der Sprache C keinerlei Einschränkungen unterworfen sind. Eine Ausnahme bildet in diesem Zusammenhang lediglich die erlaubte Anzahl der gleichzeitig geöffneten Dateien seitens MS-DOS.
Das Eröffnen einer Datei erfolgt über die Funktion »fopen«. Hierzu geben Sie den Namen der Datei sowie den erforderlichen Eröffnungsmodus an. Eine Liste der verfügbaren Modi finden Sie in Tabelle 1. In der Praxis kann dies dann so aussehen:

```
#include <stdio.h>
main(){
FILE *fp;
fp=fopen("kunden.dat","+r");
...
```

Mit Hilfe des Parameters » + r« wird die Datei »kunden.dat« zum Lesen und Schreiben geöffnet. Ist erforderlich, daß nur ein Leserecht besteht, so muß statt des Modus » + r« der Buchstabe »r« eingetippt werden. Um die Informationen der Dateieröffnung weiter benutzen zu können, wird das Ergebnis der Fopen-Funktion in der Variablen »fp« von »file« abgelegt. Alle folgenden Dateioperationen können und müssen von nun an mit dieser speziellen Variablen durchgeführt werden.
Wichtig bei jedem Schreib- oder Lesezugriff ist der aktuelle Stand des Dateizeigers. Dieser Zeiger steht immer an der Stelle, auf die als nächstes zugegriffen wird. Bei der Dateieröffnung ist die Postion meist Null.
Bei jedem Schreiben auf die Datei wird der Dateizeiger um eine Stelle in Richtung Dateiende verschoben. Sinn und Zweck jeder Dateiverwaltung ist es, Daten in einer Datei abzulegen. Eine Möglichkeit, um die Datenspeicherung zu organisieren, bietet die Funktion »fprintf«. Diese Standardroutine arbeitet analog zu »printf«. Das heißt, beliebige Texte und Werte können in formatierter Weise an die Datei weitergeleitet werden.

```
fprintf(fp,"DOS");
```

In obigem Beispiel wird der Text »DOS« in die Datei »kunden.dat« geschrieben.
Eine weitere Funktion, »fwrite«, erfüllt diese Anforderung ebenfalls. Im Gegensatz zu »fprintf« wird auf die Formatierung verzichtet. Anhand eines Beispiels werden die Unterschiede schnell sichtbar:

```
int tab[10];
main(){
FILE *fp;
fp=fopen("kd.dat","+r");
fwrite((char*),sizeof(long),
10,fp);
...
```

Diese Möglichkeit des Beschreibens einer Datei wird meist in Verbindung mit Dateien verwendet. Obiger Programmausschnitt schreibt die Tabelle »tab« in gepackter Form in die Datei. Ein Element belegt in der Datei somit zwei Byte. Verwendet man dagegen die Fprintf-Funktion, so wird jedes Zeichen innerhalb eines Elements separat (als ein Byte) geschrieben.

```
int wert=1000;
fprintf(fp,"%d",wert);
```

Geben Sie diese Datei mit dem MS-DOS-Befehl »type« aus, so erkennen Sie jedes einzelne Zeichen. Der mit »fprintf« geschriebene Wert 1000 wird genauso am Bildschrim anzeigt. Anders bei Eingabe mit »fwrite«. Hier wird nun die verstümmelte Ausgabe » ^KΦ « ausgegeben. Derartige Speicherungsunterschiede sind in bezug auf das Lesen der Datei sehr wichtig. Wird ein numerischer Wert mit »printf« geschrieben, so läßt er sich nur mit »fscanf« lesen.

```
int wert;
fscanf(fp,"%d",&wert);
```

Gleiches gilt natürlich auch für »fwrite«. Das Gegenstück hierzu lautet »fread«. Die genaue Syntax entnehmen Sie bitte Ihrem Compiler-Handbuch.
Bisher galt beim Programmieren die Annahme, daß die jeweiligen Dateifunktionen die Werte oder Texte schon richtig plazieren würden. Doch diese Annahme trifft bezüglich der Programmiersprache C nicht zu. Wie bereits in Verbindung mit Dateieröffnungen erwähnt, wird der Dateizeiger mit jedem Schreib- beziehungsweise Lesevorgang verschoben. Eine Möglichkeit, um diese Schreibposition festzulegen, bietet die in der C-Bibliothek enthaltene Funktion »fseek«. Ein Aufruf sieht etwa folgendermaßen aus:

```
fseek(fp,0L,SEEK_SET);
```

Die hier verwendete Form des Aufrufs »fseek« (siehe auch Tabelle 1) setzt den Dateizeiger auf die Position Null. Dies bewirkt die Zahl 0 in Verbindung mit der Konstanten »SEEK_SET« (Dateibeginn). Somit ist sichergestellt, daß der nächste Zugriff genau am Dateibeginn durchgeführt wird. Soll dagegen der nächste Zugriff am Dateiende erfolgen, so müssen Sie die Konstante »SEEK_SET« durch »SEEK_END« ersetzen. Alle anderen Parameter bleiben unverändert.
Um zu überprüfen, ob der Dateizeiger wirklich an der gewünschten Position plaziert wurde, kann die Funktion »ftell« aufgerufen werden:

```
long absolut;
absolut=ftell(fp);
```

Steht nach dem Aufruf zum Beispiel 190 in der Variablen »Absolut«, so bedeutet dies, daß der Dateizeiger genau auf der Stelle 190 einer Datei steht. Am Ende jeder Dateibearbeitung angelangt, ist es erforderlich, die Datei wieder zu schließen. Dies erledigt die Funktion »fclose«:

```
fclose(fp);
```

Als Parameter wird nur der jeweilige Filepointer übergeben. Sollen alle geöffneten Dateien geschlossen werden, müssen Sie die Fclose-Routine nicht in mehrmaliger Folge aufrufen. Mittels »fcloseall()« werden alle Dateien gleichzeitig geschlossen.
Ein kleines Programm hierzu, mit dem Sie mühelos mehrere Kundeninformationen bearbeiten können, zeigt das Listing.

(Harald Krause/hi)

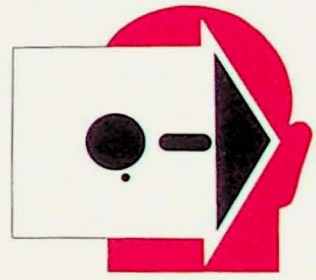
```
[FX3]   1: /************************************/
[XR5]   2: /* Kundenverwaltungs-Programm       */
[US4]   3: /*    für MS-Quick-C                */
[FX3]   4: /************************************/
[ØØØ]   5:
[L13]   6: #include <stdio.h>
[9Ø3]   7: #include <float.h>
[F14]   8: #define DATEI "kunden.dat"
[ØØØ]   9:
[RG3]  1Ø: struct kundenstamm {
[312]  11:   char loesch;
[EK1]  12:   int kdnr;
[D92]  13:   char name[2Ø];
[UV2]  14:   char strasse[2Ø];
[W22]  15:   char ort[2Ø];
[562]  16:   double summe;
[KC1]  17:  } kunde;
[ØØØ]  18:
[HS2]  19: FILE *fp,*fopen();
[G12]  2Ø: char buf[1Ø];
[ØØØ]  21:
[XV2]  22: datei_fehler(){
```

```
[6B6]  23:  printf("Dateifehler !!! -Programmende");
[C61]  24:  exit(1);
[U7Ø]  25: }
[ØØØ]  26:
[ØØØ]  27:
[VB6]  28: /* Eingabe neuer Kundeninformationen */
[ØØØ]  29:
[HP2]  3Ø: eingabe_kunde(){
[LX1]  31:   char *gets();
[B82]  32:   double atof();
[V33]  33:   fp=fopen(DATEI,"a+");
[ZT2]  34:   kunde.loesch='n';
[ØM4]  35:   printf("\nKundennummer: ");
[6N4]  36:   kunde.kdnr=atoi(gets(buf));
[YG5]  37:   printf("Name: ");  gets(kunde.name);
[4T6]  38:   printf("Strasse: "); gets(kunde.strasse);
[G95]  39:   printf("Ort: "); gets(kunde.ort);
[WZ4]  4Ø:   printf("Bestellsumme (max.): ");
[6M4]  41:   kunde.summe=atof(gets(buf));
[3L5]  42:   fwrite(&kunde,sizeof(kunde),1,fp);
[PT1]  43:   fclose(fp);
[U7Ø]  44: }
```

```
[ØØØ]   45:
[295]   46: /* Anzeigen Kundeninformation */
[ØØØ]   47:
[ØØØ]   48:
[XL2]   49: zeige_kunde(){
[5N6]   5Ø:   printf("\nKundennummer: %d",kunde.kdnr);
[D55]   51:   printf("\nName: %s",kunde.name);
[ZA6]   52:   printf("\nStrasse: %s",kunde.strasse);
[TU4]   53:   printf("\tOrt: %s",kunde.ort);
[P88]   54:   printf("\nBestellsumme (max.): %6.2f",kun
          de.summe);
[U7Ø]   55: }
[ØØØ]   56:
[8M2]   57: suche_kunde(){
[AB1]   58: int nr;
[U13]   59: fp=fopen(DATEI,"r");
[7H4]   6Ø: if(fp==NULL) datei_fehler();
[TX5]   61: printf("\nKundennummer eingeben: ");
[H83]   62: nr=atoi(gets(buf));
[SN2]   63: while(!feof(fp)){
[995]   64:   fread(&kunde,sizeof(kunde),1,fp);
[TB6]   65:   if(kunde.kdnr==nr && kunde.loesch=='n') {
[XD2]   66:       zeige_kunde();
[PT1]   67:       fclose(fp);
[KC1]   68:       return;
[U7Ø]   69:   }
[U7Ø]   7Ø:  }
[ZS5]   71:  printf("\nKunde nicht gefunden !!!");
[PT1]   72:  fclose(fp);
[U7Ø]   73: }
[ØØØ]   74:
[C18]   75: /* Logisches löschen einer Kundeninformatio
          n */
[ØØØ]   76:
[123]   77: kunde_loeschen(){
[AB1]   78:   int nr;
[XT2]   79:   long pos,ftell();
[X43]   8Ø:   fp=fopen(DATEI,"r+");
[7H4]   81:   if(fp==NULL) datei_fehler();
[TX5]   82:   printf("\nKundennummer eingeben: ");
[H83]   83:   nr=atoi(gets(buf));
[SN2]   84:   while(!feof(fp)){
[TB2]   85:     pos=ftell(fp);
[995]   86:     fread(&kunde,sizeof(kunde),1,fp);
[TB6]   87:     if(kunde.kdnr==nr && kunde.loesch=='n'
          ) {
[FT2]   88:         kunde.loesch='j';
[814]   89:         fseek(fp,pos,SEEK_SET);
[3L5]   9Ø:     fwrite(&kunde,sizeof(kunde),1,fp);
[EX5]   91:         printf("\nKunde wurde gelöscht!");
[PT1]   92:         fclose(fp);
[KC1]   93:         return;
[U7Ø]   94:       }
[7Y5]   95:    printf("\nKunde nicht vorhanden!!!");
[U7Ø]   96:   }
[U7Ø]   97: }
[4Z2]   98: /* Menue anzeigen */
[ØØØ]   99:
[113]  1ØØ: anzeigen_menue(){
[XY4]  1Ø1:   printf("\n\n\t*** Menue ***\n");
[1S4]  1Ø2:   printf("\n\t1=Neuer Kunde");
[HY3]  1Ø3:   printf("\n\t2=Suchen");
[524]  1Ø4:   printf("\n\t3=Löschen");
[1D3]  1Ø5:   printf("\n\tØ=Ende");
[RK4]  1Ø6:   printf("\n\n\tIhre Wahl: ");
[U7Ø]  1Ø7: }
[ØØØ]  1Ø8:
[ØØØ]  1Ø9:
[ØØØ]  11Ø:
[6C1]  111: main(){
[UF2]  112:   int auswahl=9;
[A13]  113:   while(auswahl!=Ø){
[1X2]  114:       anzeigen_menue();
[WY3]  115:       auswahl=atoi(gets(buf));
[NT2]  116:       switch(auswahl){
[815]  117:         case 1: eingabe_kunde(); break;
[UL4]  118:     case 2: suche_kunde(); break;
[A75]  119:     case 3: kunde_loeschen(); break;
[VB2]  12Ø:     case Ø: exit(Ø);
[356]  121:     default: printf("\nFalsche Eingabe");
[U7Ø]  122:        }
[U7Ø]  123:   }
[U7Ø]  124:  }
```

Listing. Dateiverarbeitung in C – eine einfache Sache

Sesam öffne dich!



BCi DisAsm erschließt jedes Programm!

Wandelt den Hexdump in ein dokumentiertes Assemblerlisting um und kann nach Modifizierung oder Änderung durch einen Editor mit gängigen Assemblern oder Debuggern wieder assembliert werden.

BCi DisAsm – Es gibt kaum einen, der ihm das Wasser reichen kann!

Auch Experten kennen nicht immer die Lösung, aber sie wissen, wo sie suchen müssen.
BCi DisAsm hilft JEDEM, die verwirrenden und frustrierenden Software-Puzzle, welche sich beim Umgang mit dem Computer ergeben, zu lösen. Und zwar schneller als mit jedem anderen Produkt, das Sie erwerben können, egal welche Preisklasse Sie betrachten.
Mit dem BCi DisAsm erwerben Sie Know-How, das Sie sonst Jahre an Erfahrung kosten würde. Versuchen Sie nicht, das Rad neu zu erfinden!

BCi DisAsm – Der intelligente Disassembler
BCi DisAsm ist ein intelligenter, selbsttätig dokumentierender MS-DOS Disassembler. Die ausgefeilten Routinen im BCi DisAsm analysieren in kürzester Zeit selbst komplexen und umfangreichen Programmcode und zeigen dabei potentielle Problempunkte auf.

BCi DisAsm – Das Lernmittel
Sie brauchen kein Spezialist in Assembler-Programmierung zu sein. BCi DisAsm produziert aus jedem beliebigen Programm ihres IBM-kompatiblen Computers ein gut dokumentiertes, leicht verständliches Assemblerlisting.

BCi DisAsm – Einfach in der Handhabung
BCi DisAsm kann sowohl über Parameter von der DOS-Kommandozeile aus als auch interaktiv über Menues gesteuert werden. Die Menuestruktur ist Lotus-kompatibel, selbstverständlich in Deutsch. Es besteht keine Notwendigkeit, neue Befehle zu lernen oder ständig im Handbuch nachzuschlagen. BCi DisAsm ist unmittelbar anwendbar!

Minimale Systemanforderungen
256 kB RAM, 8088/8086/80186/80286 oder 80386 CPU,
MS-DOS oder PC-DOS Version 2.0 oder größer

BCi DisAsm wird Ihnen bei den folgenden Schritten helfen:

1. Lernen Sie Assembler-Programmierung, wenn Sie möchten.
2. Entdecken Sie, warum Programm XYZ auf ihrem Rechner nicht läuft, und tun Sie etwas dagegen.
3. Ändern oder entfernen Sie Programmteile.
4. Sichern Sie sich gegen destruktive Programme ab. Finden Sie einen "Virus", und entfernen Sie ihn.
5. Integrieren Sie Code aus compilierten Programmen in eigene Anwendungen.
6. Erhöhen Sie die Kompatibilität Ihrer Software.
7. Passen Sie Software an verschiedene MS-DOS-Versionen an.

© 1987 BCi, West Germany und USA
Lotus ist eingetragenes Warenzeichen der Lotus Corp.
MS-DOS ist eingetragenes Warenzeichen der Microsoft Corp.

BCi DisAsm kostet inkl. eines sehr ausführlichen deutschen Handbuchs (mehr als 100 Seiten) nur

199,– DM*

Best.-Nr. 227 5 1/4"-Disk.
Best.-Nr. 233 3 1/2"-Disk.

199,- DM (unverbindlich Preisempfehlung)
Wenn Sie über den DMV-Bestellservice bestellen, gilt folgendes:

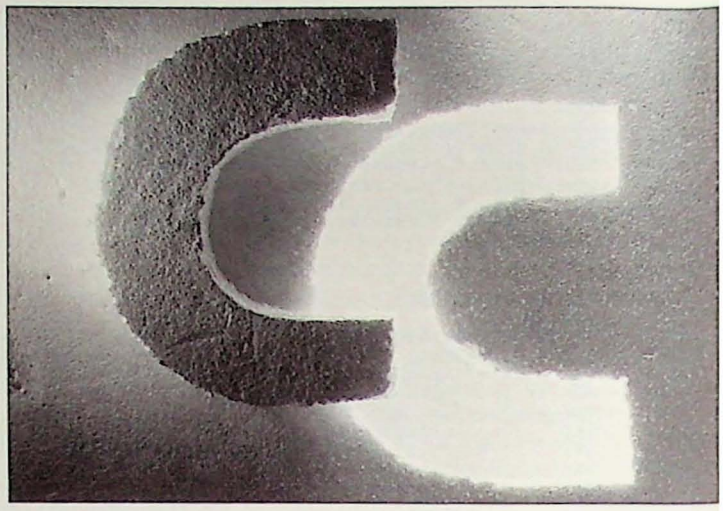
Inland:		Ausland:	
Einzelpreis	199,– DM	Einzelpreis	199,– DM
zzgl. Versandkosten	4,– DM	zzgl. Versandkosten	6,– DM
Endpreis	**203,– DM**	**Endpreis**	**205,– DM**

Händleranfragen willkommen

Bestellen Sie noch heute mittels der entsprechenden Postkarte bei:

DMV-Verlag, Postfach 250, 3440 Eschwege

Daten im Paket



In komplexen Anwendungen kommt es sehr leicht vor, daß die vorhandenen Datentypen nicht mehr genügen. Speziell der Ruf nach einem Datentyp gleich Pascals »record« wird laut – C antwortet mit »struct«.

Kombinierte Datentypen treten meist dann auf, wenn mehrere Informationen unter einem Oberbegriff verwaltet werden. Hauptanwendungsgebiet ist hierbei die Datenverwaltung.
Bei einem Bibliotheksprogramm beispielsweise ist es notwendig, verschiedene Daten über jedes vorhandene Buch zu speichern. Hierbei müssen – Buchtitel – ISBN-Nummer – Verlag – Autor – Preis etc. verwaltet werden. Programmtechnisch müßte man hierzu – sofern keine kombinierten Datentypen vorhanden wären – folgende Definitionen vornehmen:

```
char buchtitel[80];
char isbn[10];
char verlag[20];
char autor[30];
double preis;
```

Da im Normalfall zwischen Ein- und Ausbereich unterschieden wird, muß diese Definition ein zweites Mal folgen. Allein bei der Variablenbezeichnung wird es schon unklar. Stellen Sie sich aber vor, daß Daten nun von dem Eingabe- in den Ausgabebereich übertragen werden müssen – ein irrwitziger Aufwand.
Abhilfe schaffen hierfür die sogenannten »Strukturen«.
Das allgemeine Format hierzu lautet:

```
struct [Strukturname] {
Definitionen
} Variablenname;
```

Da der »Strukturname« in eckigen Klammern angegeben ist, können Sie auf die Namensvergabe verzichten. Denken Sie aber bitte an die Lesbarkeit. Um den Bibliotheks-Datensatz zu vereinbaren, läßt sich nachstehende Definition verwenden:

```
struct buchdatensatz
{
char buchtitel[80];
char isbn[10];
char verlag[20];
char autor[30];
double preis;
}buch;
```

Wie Sie sehen, werden verschiedene Datentypen innerhalb der geschweiften Klammer (char, double) angegeben. Die Struktur als solche besitzt dagegen keine feste Typvereinbarung. Um die Struktur beziehungsweise einzelne Elemente ansprechen zu können, müssen Sie zusätzlich einen symbolischen Namen (Variablenname) vergeben. Hierzu stehen zwei gleichwertige Varianten zur Verfügung. Zum einen kann dieser Name direkt bei der Definition (siehe oben) oder erst zu einem späteren Zeitpunkt vergeben werden. Eine nachträgliche Vergabe sieht dann folgendermaßen aus:

```
struct buchdatensatz buch;
```

Selbstverständlich ist es erlaubt, eine Struktur unter verschiedenen Namen anzusprechen. So bewirkt beispielsweise folgende Programmierzeile, daß die Struktur sowohl mit »ein_buch« als auch mit »aus_buch« angesprochen werden kann.

```
struct buchdatensatz ein_buch,aus_buch;
```

Diese doppelte Angabe erweist sich bei einer Trennung zwischen Ein- und Ausgabebereich als sinnvoll. Um ein einzelnes Element einer so vereinbarten Struktur anzusprechen oder zu belegen, müssen Sie die Syntax genau beachten. Das allgemeine Format hierzu lautet:

```
Strukturname.Variablenname
```

Soll jetzt ein Datensatz angelegt werden, kann etwa so vorgegangen werden:

```
strcpy(buch.buchtitel,"DOS-Extra");
strcpy(buch.isbn,"XYZ-123");
strcpy(buch.verlag,"DMV");
strcpy(buch.autor,"Goethe");
buch.preis=18;
```

Die einzelnen Operationen entsprechen genau den Zuweisungen an einzelne Variablen. Sicherlich möchten Sie die eingegebenen Daten auch irgendwann wieder zu Gesicht bekommen. Auch hier verfahren Sie analog zu »einfachen« Variablen. Soll eine Zeichenkette ausgegeben werden, so tippen Sie:

```
printf("%s",buch.buchtitel);
```

oder auch

```
puts (buch.buchtitel);
```

Als Weiterentwicklung der kombinierten Datentypen ist es auch möglich, innerhalb von Strukturen neue Strukturen einzubinden. In Listing 1 sehen Sie, daß die Variable »Autor« in einer eigenständigen Struktur untergebracht wurde. Dies kann vernünftig sein, wenn weitere Daten des jeweiligen Autors vorliegen. Um wieder ein einziges »Datenpaket« zu erhalten, lassen sich die gesplitteten Strukturen einfach zusammenfassen.

Auch Strukturen können kombinierte Datentypen beinhalten

Dies ist hauptsächlich dann interessant, wenn gleiche Daten in verschiedenen Zusammenhängen benötigt werden. Denken Sie hierbei an eine Autorendatei und eine Verlagsdatei – zumindest der Name und die Anschrift sind meist vom gleichen Typ und auch gleich groß. Beachten Sie jedoch, daß die Verschachtelungstiefe begrenzt ist. Meist bricht der Compiler nach etwa 15 Stufen ab.

Neben dieser Verschachtelung von Strukturen kennt C noch einen weiteren Datentyp – die »union« (Verbund). Aufgabe einer »union« ist es, mehrere Strukturen unter einem Oberbegriff zu verbinden. Ein praktisches Beispiel finden Sie in Ihrer Include-Datei »dos.h«. Hier werden in einer speziellen Anweisungsfolge verschiedene Registerpaare zusammengefaßt. Betrachten Sie nachfolgende Deklaration:

```
struct WORDS{
unsigned int a,b,c,d;}
struct BYTES{
unsigned int e,f,g,h;}
union REGISTER{
struct WORDS abc;
struct BYTES efg;
}reg;
```

Nach der jeweiligen Vereinbarung der einzelnen Strukturen »WORDS« und »BYTES«, wurden beide innerhalb des Verbundes »union REGISTER« zusammengefaßt. Die Elemente der Strukturen lassen sich aber

trotzdem auf einfache Art ansprechen:

```
reg.abc.a=3;
```

In diesem Beispiel wird dem Element »a« in der Struktur »WORDS« innerhalb der Union »REGISTER« der Wert 3 zugewiesen. Das allgemeine Format hierzu lautet:

```
union.struktur.element
```

Sofern Sie nach diesem Schema vorgehen, kann eigentlich kein Fehler auftreten. Für die praktische Arbeit sind aber noch verschiedene andere Feinheiten sehr nützlich. Beispielsweise ist es auch möglich, eine Tabelle aus Strukturen oder Zeiger auf Strukturen zu vereinbaren oder aber beide Varianten zu kombinieren:

```
struct buchdatensatz buchtab[100];
```

Bei obiger Deklaration handelt es sich um eine Tabelle mit 100 Elementen vom Typ »struct buchdatensatz«. Die Zuweisung eines einzelnen Tabellenelementes erreichen Sie dabei auf folgende Weise:

```
strcpy(buchtab[10].buch,"DOS");
```

Diese Variante findet häufig bei Erfassung größerer Datenbestände Anwendung. Vergleichen Sie diesbezüglich das Listing 2. Dieses Programm erlaubt die Eingabe von maximal 50 Datensätzen.

Neben Strukturtabellen besteht auch die Möglichkeit, Zeiger auf Strukturen zu benutzen. Hierbei treten jedoch kleine Unterschiede zur bisherigen Verfahrensweise auf. Die Vereinbarung eines Strukturzeigers geschieht dabei auf folgende Weise:

```
struct buchdatensatz *zeiger
```

Die Variable »zeiger« ist jetzt in der Lage, die Adresse einer Struktur vom Typ »buchdatensatz« aufzunehmen; dies allerdings gleichzeitig, damit auch nur zwei Byte des Arbeitsspeichers (im Small-Modell) bereitgestellt werden. »Echte« Daten können Sie hier nicht eingeben. Erst das Allokieren (Speicherplatz anfordern) von Arbeitsspeicher bereitet die Eingabemöglichkeit vor. Dies sieht dann so aus:

```
struct buchdatensatz *zeiger;
struct buchdatensatz *malloc();
zeiger=malloc(sizeof buchdatensatz);
```

Um jetzt auf ein einzelnes Element zu »zeigen«, benötigen Sie einen neuen Operator. Dieser wird auch als »Zeigt-auf-Operator« bezeichnet.

Es handelt sich hierbei um die Zeichenfolge »->«. Die allgemeine Form lautet:

```
zeiger->element
```

Im Beispiel des Buchdatensatzes würde dann eine Zuweisung wie folgt aussehen:

```
gets(zeiger->buchtitel)
```

Ebenso gehen Sie bei der Variablenausgabe vor.

(Harald Krause/hi)

```
[TW2]    1: struct autorinfo
[P7Ø]    2: {
[VT2]    3:  char vorname[2Ø];
[ZS2]    4:  char zuname[2Ø];
[WT1]    5:  int alter;
[5F1]    6: }autor;
[ØØØ]    7:
[ØØØ]    8:
[DK3]    9: struct buchdatensatz
[P7Ø]   1Ø: {
[AE3]   11:   char buchtitel[8Ø];
[9B2]   12:   char isbn[1Ø];
[HP2]   13:   char verlag[2Ø];
[464]   14:   struct autorinfo autor;
[252]   15:   double preis;
[H82]   16:  }buchtab[5Ø];
```

Listing 1. Ein Beispiel für verschachtelte Strukturen

```
[DK3]    1: struct buchdatensatz
[P7Ø]    2: {
[AE3]    3:   char buchtitel[8Ø];
[9B2]    4:   char isbn[1Ø];
[HP2]    5:   char verlag[2Ø];
[NR2]    6:   char autor[3Ø];
```

```
[252]    7:   double preis;
[H82]    8:  }buchtab[5Ø];
[ØØØ]    9:
[UWØ]   1Ø: main()
[P7Ø]   11: {
[LR2]   12: char buffer[1Ø];
[F11]   13: int i;
[HF2]   14: for(i=Ø;i<5Ø;i++)
[P7Ø]   15:  {
[FL4]   16:  printf("Datensatz-Nr.: %d",i);
[E54]   17:  printf("\nBuchtitel\n:");
[ØX4]   18:  gets(buchtab[i].buchtitel);
[Y33]   19:  printf("\nISBN: ");
[PW3]   2Ø:  gets(buchtab[i].isbn);
[F83]   21:  printf("\nVerlag:");
[7F4]   22:  gets(buchtab[i].verlagl);
[183]   23:  printf("\nAutor:");
[X64]   24:  gets(buchtab[i].autor);
[873]   25:  printf("\nPreis: ");
[3E5]   26:  buchtab[i].preis=atof(gets(buf));
[U7Ø]   27:  }
[U7Ø]   28: }
```

Listing 2. Das Erfassungsprogramm für maximal 50 Datensätze

Anwendungen gesucht

Haben Sie eine schnelle und komfortable Textverarbeitung geschrieben? Oder eine Datenbank mit frei wählbaren Eingabemasken? Eventuell ein Zeichenprogramm? Wenn Ihr Programm mit professionellen Programmen konkurrieren kann, ist es mit Sicherheit zu umfangreich, um als Listing in der DOS International zu erscheinen. Zögern Sie deshalb nicht, schicken Sie uns unverbindlich die aktuelle Version Ihres Programms und einen Ausdruck der Dokumentation. Eventuell wird Ihr Programm der Verkaufsschlager im Rahmen der neuen DOS-Software-Schiene. Falls Sie sich vorweg informieren möchten: Zwischen 10.00 und 12.00 Uhr und zwischen 15.00 und 17.00 Uhr stehen wir Ihnen an allen Werktagen unter der Rufnummer (089) 4 61 00 10 zur Verfügung.

Schicken Sie Ihre Unterlagen an folgende Adresse:
Redaktionsbüro Everts & Hagedorn
»DOS Software«
Hans-Stießberger-Str. 3
8013 Haar/München

Ordnung im Programmsalat

Wer sich ernsthaft mit der Programmierung befaßt, wird irgendwann feststellen, daß er um eine Art Buchführung seiner Programme nicht herum kommt. Das ist besonders dann der Fall, wenn es sich um Programmkomplexe mit vielen Modulen handelt. Der folgende Beitrag stellt zwei Filterprogramme vor, die Ihnen die Verwaltung Ihrer Programm-Module erheblich erleichtern.

Die »Buchführung« größerer Programmprojekte findet in den Modulen statt, in Form eines Programmkopfes. In diesem steht zumindest das Datum der letzten Änderung und eine Versionsnummer. Zusätzlich läßt sich dort noch ein Varianteneintrag unterbringen. Um die Sache perfekt zu machen, wird auch die Änderungsgeschichte mit aufgeführt. Ein Programm, das aus vielen Modulen zusammengesetzt ist, hat natürlich auch einen Namen. Dieser wird im Programm abgelegt. Ebenso sollte das Programm auch einen Freigabe-Eintrag erhalten. Das ist zum Beispiel eine Versionsnummer. Dies alles sind Daten, die den Hersteller in die Lage versetzen, im Fehlerfalle dem geplagten Anwender schneller eine Hilfestellung zu leisten.

Mit zwei Filterprogrammen lassen sich die nötigen Arbeitsschritte automatisieren. Das erste Programm »Upk« nimmt Ihnen die notwendigen, aber sehr lästigen Verwaltungsarbeiten ab, den Dateinamen, die Versionsnummer, einen Varianteneintrag, das Datum und die Änderungsgeschichte auf dem neuesten Stand zu halten. Zusätzlich kann der Compiler mit entsprechenden Einstellungen aufgerufen werden.

Das zweite Programm »Vers« verwaltet in einem fertig gebundenen Softwarepaket die Link- und Release-Information, die ein speziell dafür geschaffenes Modul aufnimmt. Es filtert auch alle Modulinformationen heraus, die in diesem Softwarepaket stecken.

Jedes »anständige« Programm hat einen Programmkopf, dessen Einträge immer auf dem aktuellen Stand sind. Daneben ist auch die Kommentierung wichtig. Sie muß jedoch immer noch von Hand gemacht werden, da sie sich nicht automatisch erledigen läßt.

Die Daten, die in einem Programmkopf enthalten sind, lassen sich von einem kleinen Hilfsprogramm ohne viel Aufwand immer auf dem aktuellen Stand halten. Es liegt der Gedanke nahe, ein Filterprogramm zu entwerfen, das sich an bestimmten Schlüsselwörtern in der Eingabedatei orientiert und die dazugehörigen Daten auf den neuesten Stand hält. Bild 1 zeigt ein Beispiel für einen C-Programmkopf und Bild 2 eine Assemblervariante.

Was bei diesen Programmköpfen auffällt, ist der Daten-String über dem eigentlichen Kopf. Dieser Modulkopf ist für den Compiler beziehungsweise Assembler nichts weiter als »lästiger Kommentar«, für den Programmierer hingegen ein Speicherplatz, in dem er die notwendigen und wichtigen Verwaltungsinformationen hinterlegt. Wird das Modul wie gewöhnlich compiliert, so ist von diesen Informationen nichts mehr zu finden. Wenn Sie aber ein paar Bytes »spendieren«, dann lassen sich diese Informationen auch nach der Übersetzung im Modul wiederfinden.

Im vorliegenden Beispiel sind es 46 Byte, die mit der Zeichenfolge »MODHDR.:« beginnen. In dieser Zeichenkette finden sich dieselben Informationen wieder, die in optisch ansprechenderer Form auch im Programmkopf vorkommen.

Die Definition dieses Daten-Strings ist so ausgelegt, daß sie der Compiler oder Assembler im Codesegment ablegt. Der String ist dann auch im ablauffähigen Code noch vorhanden und ebenfalls lesbar. Pro Modul werden dafür 46 Byte benötigt. Die Verwaltung dieser Daten erledigt das Programm Upk. (Die Read-Only-Anweisung ist eine Spezialität des Intel-C-Compilers, beim Zorlite-C-Compiler heißt die entsprechende Anweisung »const«.)

Wie bereits erwähnt, besteht ein großes Programm oft aus sehr vielen Modulen. Da für jedes Modul ein Platz in Form des »mod__hdr« reserviert wird, ist für eine konsequente Verwaltung auch ein ähnlicher Platz für das ablauffähige Endprodukt notwendig. Ein kleines Modul (»proghdr«) bietet den Raum für diese Eintragungen. Das Modul beinhaltet den »lnk__hdr« (Link-Header), den »rel__hdr« (Release-Header) sowie eine kleine Zugriffsroutine.

Der Link-Header trägt als wesentliche Information die Namensbezeichnung, der Release-Header die Information über einen Freigabe-Eintrag, in vielen Fällen also die Versionsnummer. Der Eintrag in den »lnk__hdr« erfolgt nach dem Link-Vorgang des neuen Programms. Der Eintrag in den »rel__hdr« geschieht nach einem entsprechenden Test, wenn das »Neugeborene« für gut befunden wurde.

Das zweite Programm Vers bearbeitet die beiden Daten-Strings. Neben den Einträgen werden jeweils Datum und Uhrzeit in diese Strings mit eingetragen – daher sind diese Felder 52 Byte lang. Vers listet dazu auch noch alle im Programm enthaltenen »MODHDR.:« auf. Damit erhalten Sie eine Zusammenstellung aller eingebundenen Module auf dem Bildschirm. Auch diese Zusammenstellung wird in einer Datei abgelegt.

```
static char readonly MOD_HDR [46] =
{'MODHDR.: DUM1.C       INTEL-C   |001 25. 3.88'};
/*....|....1....|....2....|....3....|....4....|.*/
/*************************************************************************
*      P R O J E K T    X  Y  Z                                           *
*************************************************************************
*                                                                        *
* Dateiname      : DUM1.C                                                *
* Variante       : INTEL-C                                               *
* Version        : 001                                                   *
*                                                                        *
* Compiler       : CC86                                                  *
* Params         : LARGE DEBUG                                           *
*                                                                        *
* Datum          : 25. 3.88                                              *
* Autor(en)      : Manthey                    Abt.:xxxxxxxxxxxxxxxxxxxxxx*
*                                                                        *
*************************************************************************/
/*HOC*************************************************************************
*                                                                        *
* Ver-           Kurz-                                                   *
* sion Datum     zeich Aenderungen                                       *
* ---- -------- ----- ---------------------------------------------------*
* 001  25. 3.88  MTY   New Creation in C                                 *
*************************************************************************/
```

Bild 1. Beispiel für einen Programmkopf in C...

```
crgini      name        crgini
            segment     public 'CODE'
            assume      cs:crgini
;
            DB      'MODHDR.: CRGINI.ASM    INTEL-ASM |001 27.10.88'
;
;*************************************************************************
;*      P R O J E K T    X  Y  Z                                          *
;*************************************************************************
;*                                                                       *
;* Dateiname      : CRGINI.ASM                                           *
;* Variante       : INTEL-ASM                                            *
;* Version        : 001                                                  *
;*                                                                       *
;* Compiler       : ASM86                                                *
;* Params         : DEBUG                                                *
;*                                                                       *
;* Datum          : 27.10.88                                             *
;* Autor(en)      : Manthey                    Abt.:xxxxxxxxxxxxxxxxxxxxxx*
;*                                                                       *
;*************************************************************************
;/*HOC*************************************************************************
;*                                                                       *
;* Ver-           Kurz-                                                  *
;* sion Datum     zeich Aenderungen                                      *
;* ---- -------- ----- --------------------------------------------------*
;* 001  27.10.88  MTY   New Creation                                     *
;*************************************************************************
```

Bild 2. ... und in Assembler

Upk (Update-Programm-Kopf) ist das Filterprogramm, das in der Quelldatei (Eingabedatei für den Compiler) die Daten im Programmkopf und die des »mod__hdr« auf den neuesten Stand bringt. Für die Dateizugriffe verwendet es die Low-Level-Ein-/Ausgabe-Routinen.

Ein Filterprogrammm bringt die Daten des Programmkopfes auf den jeweils aktuellen Stand

Upk ist so ausgelegt, daß innerhalb des ersten Lesevorganges der Quelldatei alle relevanten Daten im internen Datenpuffer zu finden sein müssen. Das Filterprogramm führt nacheinander folgende Schritte durch:

- Einlesen der Datei »upk.cfg« und Füllen der Speicherplätze für »Editor«, »Compiler« und »Params«. Läßt sich die Datei nicht öffnen, so wird für »Editor« ein voreingestellter Name angenommen.
- Übertragen des Dateinamens aus der Aufrufzeile in den »mod__hdr« und in den Programmkopf.
- Übernehmen des aktuellen Datums von MS-DOS in den »mod__hdr« und in den Programmkopf.
- Falls der Parameter »var« angegeben wurde, erfolgt eine Übernahme der nachfolgenden Zeichenkette in den »mod__hdr« und in den Programmkopf; sonst überträgt Upk den Varianteneintrag aus dem Programmkopf in den »mod__hdr«.
- Falls der Parameter »ver« angegeben wurde, erfolgt eine Übernahme der nachfolgenden Versionsnummer in den »mod__hdr« und in den Programmkopf oder eine Erhöhung der vorhandenen Versionsnummer um eins. Wurde der Parameter nicht angegeben, überträgt Upk die Versionsnummer aus dem Programmkopf in den »mod__hdr«.
- Falls der Bereich »History of Change« (HOC) vorhanden ist, wird das aktuelle Tagesdatum bei der entsprechenden Versionsnummer eingetragen oder – bei einer neuen Version – eine Zeile mit neuer Versionsnummer und Tagesdatum eingefügt. Dabei wird nach dem Änderungslauf der Editor mit der geänderten Datei geladen, um den HOC-Eintrag zu vervollständigen.
- Falls der Parameter »c« auftritt, ruft Upk auch den Compiler auf.

Das Filterprogramm Upk legt das Ergebnis des Änderungslaufes in einer temporären Datei mit der Namenserweiterung »upk« ab. Mit »filesize« wird zuletzt kontrolliert, ob beide Dateien die gleiche Länge haben. Bei einem positiven Resultat wird die Quelldatei gelöscht und die temporäre auf den alten Dateinamen umbenannt; andernfalls bricht Upk den Programmlauf mit einer Fehlermeldung ab. Dieses Vorgehen stellt im Falle einer »Katastrophe« (Rechnerabsturz) sicher, daß es immer noch eine Datei mit dem Inhalt der Quelldatei gibt.

Übergabeparameter für einen Compilerlauf lassen sich im Programmkopf festlegen

Nach der Änderung wird der Compiler aufgerufen. Die Fixierung der Compiler-Parameter im Programmkopf ist besonders beim C-Compiler vorteilhaft, weil diese Parameter nur über die Aufrufzeile aktiviert werden können. Eine einheitliche Compilierung mehrerer Module erfordert sonst eine Stapeldatei. Bei anderen Compilern (zum Beispiel PL/M von Intel) ist die Steuerung durch Anweisungen in der Quelldatei ebenfalls realisierbar. Dies ist besonders wichtig, wenn Sie nicht das Standardübersetzungsmodell wählen oder aber bestimmte Compiler-Einstellungen benutzen.

Der Programmkopf darf auch anders aussehen als in Bild 1 und 2 dargestellt. Upk sucht nach folgenden Zeichenketten, die in den ersten 10 KBytes stehen müssen. Dabei ist die Reihenfolge (bis auf die letzten drei) beliebig:

```
MODHDR.:
* Dateiname :
* Variante :
* Version :
* Datum :
* Compiler :
* Params :
/*HOC******
/* # # # #
# # # #*/
```

Die Änderung des Tagesdatums in der HOC ist aus der Überlegung enstanden, daß eine Version erst dann als fehlerfrei gilt, wenn sie nicht mehr übersetzt werden muß. Upk sucht nach der Zeichenfolge »/*HOC******«. Das ist der Anfang der Änderungsgeschichte, der »History of Change«. Ist diese Zeichenfolge vorhanden, sucht das Filterprogramm mit der schon bekannten Versionsnummer in diesem Bereich. Findet Upk diese Versionsnummer, setzt es auch hier das aktuelle Tagesdatum ein. Ist eine Erhöhung der Version gewünscht worden, sucht das Programm nach der Füllzeile, die mit »* # # # #« beginnt und mit »# # # # *« endet. An dieser Stelle fügt Upk eine neue Zeile ein, welche die neue Version mit dem aktuellen Tagesdatum enthält. Die genaue Beschreibung der Änderung ist dann vor der Compilierung mit dem Editor nachzuholen. Dazu lädt Upk den Editor mit dem geänderten Programm nach dem Schema: Editor, Dateiname.

Die Anmerkungen zu den Suchbegriffen im Programmkopf: Die Kopfzeilen »Compiler« und »Params« enthalten die entsprechenden Angaben. Der Aufruf erfolgt nach dem Schema: Compiler, Dateiname, Params.

Die Zeilen für »Dateinname«, »Version« und »Datum« sind zwingend erforderlich; ebenfalls die Zeile »MODHDR.:«. Dagegen darf die Zeile »Variante« fehlen. Ein Compiler-Aufruf erfolgt nur, wenn dem Upk-Programm über die Zeile »Compiler« oder über den Parameter »Compiler« in der Konfigurationsdatei ein Compiler mitgeteilt wurde. Die Zeile »Params« kann dabei allerdings fehlen. Eine Bearbeitung der »History of Change« ist nur dann gewährleistet, wenn alle Suchbegriffe »/*HOC******«, »* # # # #« und »# # # # *« vorhanden sind.

»History of Change« dokumentiert die Änderungsgeschichte der Programm-Module

In der Konfigurationsdatei »upk.cfg« lassen sich drei Parameter verwenden: »Editor«, »Compiler« und »Params«. Der Eintrag für den Compiler und die Parameter sind Voreinstellungen für den Fall, daß in den Programmköpfen diese Parameter nicht vorhanden sind. Liegen sie dort jedoch vor, so haben sie Vorrang gegenüber den Voreinstellungen aus der Konfigurationsdatei. Bild 3 zeigt, wie eine solche Datei aussieht.

Für die Änderung der HOC ist ein Editor notwendig. Dieser ist über den Eintrag »Editor« zu definieren. Läßt sich die Datei »upk.cfg« nicht zum Lesen öffnen, so wird ein voreingestellter Wert eingesetzt (hier als Beispiel der Intel-Editor »Aedit«). Es darf hier jeder Editor eingesetzt werden, der eine reine ASCII-Datei erzeugt. Der Eintrag muß mit [Enter] abgeschlossen werden, weil später im Programm direkt an dieser Position der Dateiname der zu bearbeitenden Textdatei eingefügt wird.

Für einen einfachen Aufruf ist hier auch die entsprechende Anzahl von Leerzeichen einzutragen und dann erst [Enter] zu drücken. Somit läßt sich auch ein Pfad für den Textteil vorgeben (wichtig zum Beispiel bei Star-Writer mit seinen Voreinstellungen). Beispiele für Einträge: Aus

```
editor= c:\star3\writer c:\cdir\[Enter]
```

wird für »test.c«

```
C:\STAR3\WRITER C:\CDIR\TEST.C
```

```
editor= c:\star3\writer c:\ccc\
Compiler=c:\ndx\cc86 c:\ccc\
Params= LARGE DEBUG
```

Bild 3. So könnte die Konfigurationsdatei »upk.cfg« aussehen

und aus

`editor=aedit [Enter]`

wird

`AEDIT TEST.C`

Gleiches gilt ebenso für den Eintrag »Compiler« und »Params«. Bei »Params« ist jedoch mit einem führenden Zwischenraum zu beginnen. Die Konfigurationsdatei läßt sich auch zentral installieren. Dazu muß in der Stapeldatei »autoexec.bat« die Zeile

`set upk=c:\tool`

eingefügt werden, wenn »tool« das Unterverzeichnis ist, in dem sich »upk.cfg« befindet. Das Filterprogramm Upk durchsucht dann zuerst den Environment-Bereich nach »upk«. Findet es diesen Eintrag, so liest das Programm die Konfigurationsdatei aus dem dort angegebenen Verzeichnis, sonst aus dem aktuellen.
Die Aufrufvarianten für das Programm Upk:

`upk <Dateiname.Ext>`

Dateiname und Datum werden aktualisiert.

`upk <Dateiname.Ext> ver`

Erhöht zusätzlich die Versionsnummer um eins.

`upk <Dateiname.Ext> vernnn`

Übernimmt die neue Versionsnummer »nnn«. Es wird vorausgesetzt, daß diese drei Zeichen Ziffernzeichen sind. Eine Überprüfung findet nicht statt.

`upk <Dateiname.Ext> VAR...8...`

Übernimmt die nachfolgenden Zeichen (maximal acht) und trägt sie im Feld »Variante« ein. Diese Zeichen werden alle in Großbuchstaben umgewandelt.
Die Parameter »var« und »ver« (mit Anhang) dürfen Sie zusammen und in beliebiger Reihenfolge verwenden. Zusätzlich läßt sich bei allen Aufrufen der Steuerparameter »c« für Compilierung anhängen (in den Beispielen nicht angegeben).

Ein Filterprogramm verwaltet Link- und Release-Informationen

Das Programm Vers ist das Verwaltungsprogramm, das in einem gebundenen Programmpaket in die dafür bestimmten Stellen des Moduls »proghdr« den Link-Eintrag und den Freigabevermerk einsetzt. Bei beiden Einträgen, die zeitlich nacheinander erfolgen, wird zum Datum auch noch die Uhrzeit eingetragen. Im »lnk_hdr« steht es Ihnen frei, zusätzlich zum Programmnamen noch eine weitere Information mit zehn Zeichen unterzubringen. Diesen Vorgang steuert der Parameter »lnk«.
Mit dem Parameter »rel« tragen Sie den Freigabevermerk ein. Diese Information darf bis zu 23 Zeichen umfassen. Die Zeile »rel_hdr« bietet dafür den entsprechenden Platz.
Das Programm Vers sucht nach den beiden Zeichenketten »LNKHDR.:« und »RELHDR.:«, die in der angegebenen Reihenfolge stehen müssen. Sie sind wichtig für die Verwaltung. Der dritte Suchbegriff »MODHDR.:« dient nur für die Zusammmenstellung.
Soll in einer großen Datei etwas gesucht werden, so wird die Suche per Programm in Teilabschnitten durchgeführt. Diese Teilbearbeitung birgt die Gefahr in sich, daß der Suchbegriff selbst zerteilt wird, sofern er mehr als ein Zeichen lang ist; das heißt, der erste Teil des Begriffs steht an letzter Stelle des internen Speichers, während der zweite Teil beim nächsten Lesezugriff am Anfang steht. In diesem Falle ist der Suchbegriff per Programm nicht auffindbar. Sollte dieser Fall bei »LNKHDR.:« eintreten, so bricht Vers den Programmlauf mit einer Fehlermeldung ab. Tritt die gleiche Situation bei »MODHDR.:« auf, so fehlt diese Zeile für das betreffende Modul in der Aufstellung. Dem Programm Vers ist zwar nicht bekannt, wie viele Module im Programmpaket enthalten sind, es muß aber in jedem Fall »lnkhdr« vorhanden sein.

Für eine Zerteilung der Zeichenkette des »modhdr«, des »lnkhdr« und des »relhdr« ist dagegen Vorsorge getroffen. Vers läuft in drei Schleifen ab. Die erste prüft, ob der Suchbegriff »LNKHDR.:« in der Datei vorhanden ist und ob noch bis zum Buffer- oder Datei-Ende 128 Byte vorhanden sind. Das Programm erwartet, daß beide Header innerhalb von 128 Byte stehen. Diese Zahl ermittelt sich aus 2x52 Byte je Header, zuzüglich einiger Bytes für die Ausrichtung; großzügig aufgerundet sind das also 128 Byte.

Bei einem Rel- oder Lnk-Parameter müssen beide Header für die Bearbeitung zusammenhängend im Speicher stehen. Dies überprüft Vers in der ersten Schleife. Findet es den Begriff »LNKHDR.:« und sind es weniger als 128 Byte bis zum Bufferende, so ist eine korrekte Eintragung nicht möglich. Daher wird die Anzahl der zu lesenden Bytes pro Read-Aufruf für den zweiten Durchlauf um 128 reduziert. Damit ist sichergestellt, daß beim zweiten Lesen der Datei dieser benötigte Block auch tatsächlich vollständig im Speicher steht. Erst wenn es mehr als 80 Read-Aufrufe gibt (800 KByte) ist wieder eine Zerteilung denkbar.

Die zweite Schleife ist die eigentliche Bearbeitungsschleife: Hier wird die modifizierte Datei auch zurückgeschrieben, falls ein Rel- oder Lnk-Parameter gesetzt wurde. Unabhängig davon zeigt Vers in dieser Schleife den »lnk_hd« und den »rel_hdr« an, um diese beiden Header an den Anfang der Auflistung zu stellen.

Durch eine einfache Sicherungsmaßnahme reagieren Sie auf Veränderungen Ihres Programmcodes

In der dritten Schleife durchsucht das Programm die Eingabedatei noch einmal nach allen »MODHDR.:« und zeigt sie auf dem Bildschirm an. Tritt eine Teilung der Zeichenkette des »modhdr« ein, so wird dies erkannt, und beide Teile werden wieder zu einer Zeile zusammengefügt. Diesen Fall kennzeichnen zwei Plus-Zeichen am Zeilenanfang. Die Zusammenstellung aller Header speichert Vers in einer Datei mit der Erweiterung »hdr«.

Erzeugt das Programm Vers eine neue Datei infolge des Lnk- oder Rel-Parameters, so erhält diese Datei die Endung »ver«. Anschließend kontrolliert Vers mit »filesize«, ob beide Dateien die gleiche Länge besitzen. Bei gleicher Länge der Dateien wird die Quelldatei gelöscht und die temporäre auf den alten Dateinamen umbenannt; andernfalls bricht Vers die Bearbeitung mit einer Fehlermeldung ab.

Mit dem Eintrag in den »rel_hdr« läßt sich noch eine kleine Sicherungsmaßnahme verbinden: Über die Zeichen, die mit »rel« einzutragen sind, bildet Vers eine Prüfsumme. Im Programm geschieht dies, indem der Text Byte für Byte in einer Integer-Variablen addiert und das Komplement dieses Wertes um eins erhöht in die Stellen 32 und 33 des »rel_hdr« eingetragen wird. Mit den gleichen Anweisungen überprüft das neu gebundene Programm später, ob der Freigabe-Eintrag entfernt oder verändert wurde. Als Antwort auf eine solch »frevelhafte Tat« lassen Sie Ihr Programm sofort den Dienst einstellen oder in einer Endlosschleife laufen oder... Damit sich diese Maßnahmen nicht so leicht erkennen oder umgehen lassen,

sollte sich die Prüfsumme aber nicht unmittelbar hinter der Zeichenkette befinden.

Die verschiedenen Formen der Vers-Aufrufe lauten:

```
vers <Datei.Ext>
```

Vers durchsucht die Datei nach der Buchstabenfolge »LNKHDR.:« und »RELHDR.:« und stellt diese Daten an den Anfang der Auflistung; danach folgen die »MODHDR.:«. Die Auflistung wird unter dem beim Aufruf angegebenen Dateinamen mit der Erweiterung »hdr« abgelegt.

```
vers <Datei.Ext> lnk...10...
```

Vers trägt in das Feld »lnk__hdr« den Dateinamen ein und fügt Tagesdatum und Uhrzeit selbständig ein. Zusätzlich ist nach dem Dateinamen ein Eintrag mit maximal 10 Zeichen erlaubt. Ein eventueller Eintrag im Feld »rel__hdr« wird gelöscht. Anschließend erfolgt die Auflistung der vorhandenen »MODHDR.:«.

```
vers <Datei.Ext> rel...23...
```

Vers trägt in das Feld »rel__hdr« die dem Parameter »rel« mitgegebene Zeichenkette ein (maximal 23 Zeichen). Tagesdatum und Uhrzeit werden selbständig angefügt. Anschließend erfolgt die Auflistung aller vorhandenen »MODHDR.:«.

(Reinhard Manthey/zi)

Name	upk.c, vers.c
Sprache	Zorlite C
Grafikkarte	jede
Besonderheiten	keine

```
[RP7]    1: /*****"MODHDR.: UPK.C        ZORLITE   |ØØ2
             15. 2.89"********
[G76]    2: ********************************************
            ******************
[396]    3: * Dateiname      : UPK.C
                              *
[YE6]    4: * Variante       : ZORLITE
                              *
[TR5]    5: * Version        : ØØ2
                              *
[SX5]    6: * Compiler       : ccl
                              *
[ES5]    7: * Datum           : 15. 2.89
                              *
[P39]    8: * Autor(en)      : R.Manthey, Gasparistr.7,
            8ØØØ Muenchen 71 *
[G76]    9: ********************************************
            ******************
[YT5]   1Ø: * Bedeutung:
                              *
[CN6]   11: *-----------------------------------------
            -----------------*
[BT9]   12: * Update-Programm-Kopf : versorgt den Modul
            kopf und den      *
[GS9]   13: * MODHDR mit Tagesdatum, Varianteneintrag
            und Versions-     *
[C58]   14: * nummer. Startet Editor und Compiler.
                              *
[8T4]   15: *
                              *
[XB6]   16: ********************************************
            ******************/
[L13]   17: #include <stdio.h>
[6A3]   18: #include <string.h>
[UR2]   19: #include <time.h>
[9L2]   2Ø: #include <dos.h>
[ØØØ]   21:
[ØØØ]   22:
[5U3]   23: #define  BUFSIZE 1Ø * 1Ø24
[RR3]   24: /*   Puffergroesse          */
[P83]   25: #define  PMODE          Ø
[KP3]   26: #define  O_RDONLY       Ø
[EM3]   27: #define  NULL    (char *)Ø
[ØØØ]   28:
[PØ5]   29: /*------- Globale Variable ---------*/
[5Y2]   3Ø: int        fd1,fd2,fd3;
[145]   31: /* Dateidescriptoren  E / A          */
[FG2]   32: int        n_read ;
[TD5]   33: /* Anzahl gelesener Zeichen          */
[GV2]   34: int        n_line = Ø;
[G85]   35: /* Laenge der neuen HOC-Line         */
[9Z1]   36: int        i= Ø;
[VK5]   37: /* Zaehlvariable fuer Schleife       */
[ØØØ]   38:
[6A3]   39: char       hdr_flg          ;
[RU5]   4Ø: /* Bearbeitungsflag fuer Header      */
[VB3]   41: char       var_flg          ;
[EZ5]   42: /* Bearbeitungsflag fuer Variante    */
[ZB3]   43: char       ver_flg          ;
[YY5]   44: /* Bearbeitungsflag fuer Version     */
[L93]   45: char       hoc_flg          ;
[A56]   46: /* Bearbeitungsflag fuer HOC malloc  */
[DT3]   47: char       com_flg = ØxFF   ;
[UA4]   48: /* Compilerflag                      */
[XU3]   49: char       par_flg = ØxFF   ;
[1L5]   5Ø: /* Parameterflag fuer Compiler       */
[ØØØ]   51:
[M53]   52: char       *m_ptr           ;
[BF4]   53: /* Zeiger auf MODHDR                 */
[T53]   54: char       *r_ptr           ;
```

```
[ZA5]   55: /* Zeiger auf einzelne Zeichen       */
[ØØØ]   56:
[X53]   57: char       *v_ptr           ;
[ZA5]   58: /* Zeiger auf einzelne Zeichen       */
[F53]   59: char       *h_ptr           ;
[M54]   6Ø: /* Hilfszeiger                       */
[MC3]   61: char       *mal_ptr         ;
[V65]   62: /* Hilfszeiger fuer malloc           */
[ØØØ]   63:
[532]   64: char       *path;
[4F5]   65: /* Zeiger auf Environment UPK=       */
[GW2]   66: char       *date_ptr;
[Z15]   67: /* Zeiger auf Zeitstring             */
[ØØØ]   68:
[BA3]   69: char       ver_buf[1Ø];
[LN5]   7Ø: /* Zwischenpuffer fuer Version       */
[L93]   71: char       par_buf[2Ø];
[5T5]   72: /* Zwischenpuffer fuer Parameter     */
[S83]   73: char       wrkfile[2Ø];
[EK5]   74: /* Dateiname ohne Ext. f. Compiler   */
[Z73]   75: char       newfile[2Ø];
[3N5]   76: /* Dateiname fuer neue Zieldatei     */
[283]   77: char       comline[99];
[EØ5]   78: /* Commandline fuer DOS              */
[EL3]   79: char       compiler[6Ø];
[AV4]   8Ø: /* Compilermerkzeile                 */
[EZ2]   81: char       params[6Ø];
[856]   82: /* Parametermerkzeile fuer Compiler  */
[193]   83: char       act_dir[6Ø];
[FØ5]   84: /* aktuelles Directory               */
[LZ2]   85: char       editor[6Ø];
[AL5]   86: /* Editor fuer Aenderung der HOC     */
[F84]   87: char       r_buf [BUFSIZE]  ;
[WL3]   88: /* Buffer                            */
[384]   89: /*.................................*/
[K52]   9Ø: struct tm *bd;
[ØG1]   91: time_t t;
[ZG5]   92: /*--- Globale Variable ---- E n d e ---*/
[S64]   93: extern char* malloc(int);
[AW3]   94: extern void free(char*);
[XG4]   95: extern char* getenv(char*);
[AF4]   96: /*-------------------------------------*/
[GN2]   97: int error (s1,s2)
[AF4]   98: /*-------------------------------------*/
[M22]   99: char *s1 , *s2 ;
[P7Ø]  1ØØ: {
[H92]  1Ø1: printf (s1,s2);
[Z62]  1Ø2: printf ("\n");
[591]  1Ø3: exit (1);
[U7Ø]  1Ø4: }
[ØØØ]  1Ø5:
[AF4]  1Ø6: /*-------------------------------------*/
[W82]  1Ø7: int comp (m,s)
[446]  1Ø8: /* vergleicht Zeichenstring mit       */
[WL4]  1Ø9: /* Memorywerten                      */
[AF4]  11Ø: /*-------------------------------------*/
[171]  111: char *m;
[ØH5]  112: /* Pointer auf einen Zeichenstring
[3B4]  113:    im Memory ohne \Ø                  */
[T71]  114: char *s;
[GT5]  115: /* Pointer auf Characterstring mit \Ø
[9B4]  116:    abgeschlossen                      */
[P7Ø]  117: {
[5F3]  118:     for ( ; *s == *m; s++, m++)
[M92]  119:         if (*(s+1) == '\Ø')
[NS1]  12Ø:             return (Ø);
[H12]  121:             /* gefunden */
[TE2]  122:         return (*s - *m);
```

Listing 1. »upk.c« nimmt Ihnen Verwaltungsarbeit bei der modularen Programmierung ab

```
[4X2]  123:          /* nicht gefunden */
[U7Ø]  124: }
[ØØØ]  125:
[AF4]  126: /*-------------------------------------*/
[GN3]  127: char *search (ms,me,cs)
[WA6]  128: /* sucht in einem Memorybereich nach   */
[N45]  129: /* einem Zeichenstring                 */
[AF4]  13Ø: /*-------------------------------------*/
[FB1]  131: char *ms;
[LD5]  132: /* Startpointer im Memory              */
[1B1]  133: char *me;
[C35]  134: /* Endpointer   im Memory              */
[7A1]  135: char *cs;
[9C6]  136: /* Referenzpointer auf Characterstring
[EL3]  137:    mit \Ø                               */
[P7Ø]  138: {
[NM3]  139: while (comp (ms,cs) != Ø)
[P7Ø]  14Ø:     {
[AG4]  141:     if (++ms == me) return (NULL);
[4X2]  142:     /* nicht gefunden */
[U7Ø]  143:     }
[ØY1]  144: return (ms);
[H12]  145: /* gefunden */
[U7Ø]  146: }
[ØØØ]  147:
[ØØØ]  148:
[AF4]  149: /*-------------------------------------*/
[XA2]  15Ø: char *date(ptm)
[AF4]  151: /*-------------------------------------*/
[2G2]  152: struct tm *ptm;
[P7Ø]  153: {
[LS5]  154:  static char d_line[9] = "dd.mm.yy";
[ØØØ]  155:
[RE5]  156:  d_line[Ø] = (ptm->tm_mday >= 1Ø) ?
[574]  157:     (ptm->tm_mday / 1Ø) + 'Ø' : ' ';
[NP5]  158:  d_line[1] = (ptm->tm_mday % 1Ø) + 'Ø';
[ØØØ]  159:
[NA5]  16Ø:  d_line[3] = (ptm->tm_mon >= 1Ø) ?
[L44]  161:     (ptm->tm_mon / 1Ø) + 'Ø' : ' ';
[GG5]  162:  d_line[4] = (ptm->tm_mon % 1Ø) + '1';
[ØØØ]  163:
[7T5]  164:  d_line[6]  = (ptm->tm_year / 1Ø) + 'Ø';
[YS5]  165:  d_line[7]  = (ptm->tm_year % 1Ø) + 'Ø';
[ØØØ]  166:
[TD2]  167:  return d_line;
[U7Ø]  168: }
[AF4]  169: /*-------------------------------------*/
[ZG2]  17Ø: main (argc,argv)
[AF4]  171: /*-------------------------------------*/
[532]  172: int       argc;
[RE2]  173: char *    argv[];
[ØØØ]  174:
[P7Ø]  175: {
[AF4]  176: /*-------------------------------------*/
[4W1]  177: hdr_flg = Ø;
[RW1]  178: var_flg = Ø;
[3X1]  179: ver_flg = Ø;
[ØØØ]  18Ø:
[DD1]  181: time(&t);
[LK3]  182: /* get current time     */
[DZ2]  183: bd = localtime(&t);
[2B3]  184: date_ptr = date (bd);
[ØØØ]  185:
[ØY7]  186: printf ("-----------------------------------
           ---------------------\n");
[XV7]  187: printf ("UPK  V 1.3  Update - Programm - Ko
           pf\n");
[3G7]  188: printf ("(C)RaMtY              Datum: %s\n",
           date_ptr);
[ØY7]  189: printf ("-----------------------------------
           ---------------------\n");
[ØØØ]  19Ø:
[AF4]  191: /*-------------------------------------*/
[ØØØ]  192:
[VP1]  193: if (argc < 2)
[P7Ø]  194:  {
[XL7]  195:    puts ("\nAnzahl der Argumente nicht ok !!
           !\n");
[MHB]  196:    printf ("Aufruf :\nUPK  <Datei> {VERnnn}
           {VARxxxxxxxx} {C}\n");
[591]  197:    exit (1);
[U7Ø]  198:  }
[ØØØ]  199:
[AF4]  2ØØ: /*-------------------------------------*/
[UT2]  2Ø1: strupr(argv[1]);
[H85]  2Ø2: /* Dateinamen in Grossbuchstaben
[9G3]  2Ø3:    umwandeln                          */
[WT2]  2Ø4: strupr(argv[2]);
[575]  2Ø5: /* Parameter1 in Grossbuchstaben
[9G3]  2Ø6:    umwandeln                          */
[YT2]  2Ø7: strupr(argv[3]);
[775]  2Ø8: /* Parameter2 in Grossbuchstaben
[9G3]  2Ø9:    umwandeln                          */
[ØU2]  21Ø: strupr(argv[4]);
[975]  211: /* Parameter3 in Grossbuchstaben
[9G3]  212:    umwandeln                          */
[ØØØ]  213:
[615]  214: /*--- Dateinamen ohne Extension
[GM5]  215:       erzeugen fuer Compileraufruf ---*/
[H44]  216: strcpy(wrkfile,argv[1]);
[VE6]  217: if ((r_ptr = strchr(wrkfile,'.')) != NULL)
[562]  218:    *r_ptr = '\Ø';
[5C4]  219: /*-------------------------------------*/
[ØØØ]  22Ø:
[3N3]  221: /* Quelldatei oeffnen */
[XE6]  222: if (( fd1 = open (argv[1], O_RDONLY)) == -1
           )
[RX5]  223:    error ("UPK :  can't open %s",argv[1]);
[ØØØ]  224:
[ØØØ]  225:
[RS4]  226: if (argc > 2)  strupr(argv[2]);
[GM5]  227: /* Parameter in Grossbuchstaben */
[ØØØ]  228:
[TF4]  229: /* Dateinamen mit Extension
[XF3]  23Ø:    ".UPK" erzeugen            */
[K44]  231: strcpy(newfile,argv[1]);
[ØØØ]  232:
[2K6]  233: if ((r_ptr = strchr(newfile,'.')) == NULL)
[LU3]  234:    strcat(newfile,".UPK");
[UF4]  235:   else strcpy(r_ptr,".UPK");
[ØØØ]  236:
[ZH3]  237: /* Zieldatei oeffnen */
[S86]  238: if (( fd2 = creat  (newfile, PMODE   )) ==
           -1)
[B96]  239:    error ("UPK :  can't create %s",newfile)
           ;
[ØØØ]  24Ø:
[EP5]  241: /*---Configdatei oeffnen---------------*/
[R63]  242: getcwd (act_dir,59);
[984]  243: /* aktuells DIR sichern */
[ØØØ]  244:
[3D5]  245: if ((path = (getenv("UPK"))) != NULL)
[P7Ø]  246:     {
[GA4]  247: /* Path fuer UPK gefunden */
[8Z1]  248:     chdir (path);
[Z14]  249: /* neues DIR einstellen */
[U7Ø]  25Ø:     }
[ØØØ]  251:
[8D6]  252: if (( fd3 = open ("UPK.CFG", O_RDONLY)) ==
           -1)
[P7Ø]  253:     {
[GA5]  254: /* AEDIT ist Defaulteinstellung,
[Y75]  255:    wenn keine CFG-Datei vorhanden */
[BØ4]  256:     strcpy (editor,"AEDIT  ");
[U7Ø]  257:     }
[ZRØ]  258: else
[P7Ø]  259:  {
[YF3]  26Ø: /* CFG-Datei gefunden */
[ØØØ]  261:
[ME5]  262:   n_read = read (fd3,r_buf,BUFSIZE);
[8F2]  263:   strupr(r_buf);
[7L5]  264: /* alle Angaben in Grossbuchstaben
[LN3]  265:    umwandeln                          */
[ØØØ]  266:
[RA5]  267:   if ((r_ptr = search(r_buf,r_buf+
[L64]  268:        n_read,"EDITOR=")) != NULL)
[P7Ø]  269:    {
[LS1]  27Ø:     r_ptr += 7;
[RB6]  271:     if ((h_ptr = strchr(r_ptr,'\n')) != NUL
           L)
[P7Ø]  272:     {
[7L2]  273:      *(h_ptr-1) = ØxØØ;

[RT3]  274:      strcpy (editor,r_ptr);
[U7Ø]  275:     }
[348]  276:     else  error ("UPK :  missing <CR> in UK
           P.CFG",NULL);
[U7Ø]  277:    }
[GP8]  278:    else error ("UPK : EDITOR not found in U
           PK.CFG", NULL);
[ØØØ]  279:
[RA5]  28Ø:    if ((r_ptr = search(r_buf,r_buf+
[6G4]  281:         n_read,"COMPILER=")) != NULL)
[P7Ø]  282:    {
[TS1]  283:     r_ptr += 9;
[RB6]  284:     if ((h_ptr = strchr(r_ptr,'\n')) != NUL
           L)
[P7Ø]  285:     {
[7L2]  286:      *(h_ptr-1) = ØxØØ;
[A74]  287:      strcpy (compiler,r_ptr);
[7W1]  288:      com_flg = Ø;
[U7Ø]  289:     }
[U7Ø]  29Ø:    }
[ØØØ]  291:
[RA5]  292:    if ((r_ptr = search(r_buf,r_buf+
[G74]  293:         n_read,"PARAMS=")) != NULL)
[P7Ø]  294:    {
[LS1]  295:     r_ptr += 7;
[RB6]  296:     if ((h_ptr = strchr(r_ptr,'\n')) != NUL
           L)
[P7Ø]  297:     {
```

```
[7L2]  298:       *(h_ptr-1) = 0x00;
[MU3]  299:       strcpy (params,r_ptr);
[BW1]  300:       par_flg = 0;
[U70]  301:      }
[U70]  302:     }
[9P1]  303:     close (fd3);
[U70]  304:    }
[000]  305:
[5K2]  306: chdir (act_dir);
[8S5]  307: /* altes DIR wieder restaurieren */
[000]  308:
[KV5]  309: printf ("UPK : Configurations:\n");
[HE5]  310: printf ("       Editor   = %s\n",editor);
[V46]  311: printf ("       Compiler = %s\n",compiler);
[2E5]  312: printf ("       Params   = %s\n",params);
[0Y7]  313: printf ("-----------------------------------
           --------------------\n");
[000]  314:
[AF4]  315: /*--------------------------------------*/
[000]  316:
[0X6]  317: printf ("UPK : processing %s \n",argv [1]);
[000]  318:
[W85]  319: /*---- Kopierschleife  ----------------*/
[000]  320:
[264]  321: while (( n_read = read (fd1,
[W03]  322:         r_buf,BUFSIZE)) > 0)
[P70]  323:  {
[8E2]  324:   r_ptr = r_buf;
[000]  325:
[NZ4]  326:   while (r_ptr < (r_buf + n_read))
[P70]  327:   {
[EF2]  328:    if (hdr_flg == 0)
[P70]  329:    {
[4V5]  330: /* Bearbeitung des Programm-Kopfes */
[000]  331:
[U55]  332: /*--------- MODHDR suchen -------------*/
[000]  333:
[8A5]  334:     if ((m_ptr = search(r_buf,r_buf+
[Y84]  335:          n_read,"MODHDR.:")) == NULL)
[Y86]  336:          error ("UPK : MODHDR not found", N
           ULL);
[000]  337:
[ZR1]  338:             m_ptr -= 1;
[ZM4]  339: /* Stelle von MODHDR merken */
[000]  340:
[2R5]  341: /*--------- Dateinamen bearbeiten -----*/
[000]  342:
[RA5]  343:     if ((r_ptr = search(r_buf,r_buf+
[CA5]  344:          n_read,"* Dateiname     :")) == NU
           LL)
[UU6]  345:          error ("UPK : Dateiname not found"
           , NULL);
[000]  346:
[1F4]  347:          /* Dateiname in PK loeschen;
[DT3]  348:             neuen eintragen          */
[1T3]  349:          memset  (r_ptr+18,' ',15);
[NX7]  350:          strncpy (r_ptr+18, argv [1],strlen
           (argv [1]));
[000]  351:
[DS4]  352:          /* und in MODHDR uebertragen */
[H75]  353:          strncpy (m_ptr+10, r_ptr+18, 15);
[000]  354:
[NH5]  355: /*-------- Datum uebernehmen ----------*/
[000]  356:
[RA5]  357:          if ((r_ptr = search(r_buf,r_buf+
[105]  358:               n_read,"* Datum         :"))
           == NULL)
[F66]  359:               error ("UPK : Datum not found
           ", NULL);
[000]  360:
[BA5]  361:          strncpy (r_ptr+18, date_ptr ,8);
[CE4]  362: /* Datum in PK einkopieren */
[Y95]  363:          strncpy (m_ptr+38, date_ptr ,8);
[RS3]  364: /* Datum in MODHDR         */
[000]  365:
[PZ5]  366: /*--- Compiler-Eintrag uebernehmen ---*/
[000]  367:
[RA5]  368:          if ((r_ptr = search(r_buf,r_buf+
[P45]  369:               n_read,"* Compiler      :"))
           != NULL)
[P70]  370:            {
[AB5]  371:             strncpy (compiler,r_ptr+18,59);
[VT6]  372:             com_flg = 0;/* wenn Compiler vo
           m Kopf..... */
[KF2]  373:             par_flg = 0xFF;
[UU6]  374: /*... dann keine Params von der CFG-Datei*/
[U70]  375:            }
[ZR0]  376:            else
[P70]  377:            {
[9X9]  378:             printf("UPK : Compiler in Heade
           r not found... continue...\n");
[U70]  379:            }
```

Listing 1. »upk.c« nimmt Ihnen Verwaltungsarbeit bei der modularen Programmierung ab

PCpur

Das neue PC-Magazin für Einsteiger und Aufsteiger...

PCpur bietet Ihnen Heft für Heft:
- Programme: Anwendungs- und Spielprogramme in GW-, Turbo- und Quick-BASIC
- Tips und Tricks: Alles, was den Umgang mit dem Computer erleichtert; alles, was Sie schon immer wissen wollten.
- Durchblick: Hier wird das Innere Ihres PC anhand kleiner BASIC-Programme anschaulich erläutert.
- Prüfstand: Aktuelle Tests von Anwendungs- und Hilfsprogrammen der Low-Cost-Preisklasse
- Unterhaltung: Tests von Spielprogrammen, Hinweise auf Lösungswege zu kniffligen Spielen, Lustiges und Entspannendes
- Public Domain: Vorstellung und Tests von Programmen aus dem Fundus der Public-Domain- und Shareware-Vertreiber

In Heft 6/89 finden Sie:
- Zeichnen lernen auf dem PC – ein ungewöhnlicher Kurs
- Teil 2: Porträts
- Top-Programm zum Abtippen: eine komplette Kfz-Kostenverwaltung
- Ausführlicher Testbericht: Zusatzhardware für den EURO-PC
- Interviews mit je einem Shareware-Autor und -Händler
- Hardwarebasteleien: Tuning für Ihren PC

DMV-Verlag · Postfach 250 · 3440 Eschwege

DMV Daten- und Medienverlag

```
[ØØØ]  38Ø:
[AN6]  381: /*-- Params fuer Compiler uebernehmen --*/
[ØØØ]  382:
[RA5]  383:           if ((r_ptr = search(r_buf,r_buf+
[GY4]  384:                n_read,"* Params        :"))
        != NULL)
[P7Ø]  385:           {
[TW4]  386:            strncpy (params,r_ptr+18,59);
[BW1]  387:            par_flg = Ø;
[U7Ø]  388:           }
[ZRØ]  389:           else
[P7Ø]  39Ø:           {
[VF9]  391:            printf("UPK : Params in Header n
        ot found... continue...\n");
[U7Ø]  392:           }
[ØØØ]  393:
[BZ5]  394: /*--------- Variante uebernehmen -------*/
[ØØØ]  395:
[RW1]  396:          var_flg = Ø;
[ØØØ]  397:
[RA5]  398:           if ((r_ptr = search(r_buf,r_buf+
[255]  399:                n_read,"* Variante      :"))
        != NULL)
[P7Ø]  4ØØ:           {
[ØE6]  4Ø1:            if ((argc > 2) && (comp (argv [2
        ],"VAR") == Ø))
[P7Ø]  4Ø2:            {
[VW1]  4Ø3:             var_flg = 1;
[L15]  4Ø4:             strcpy (par_buf,&argv[2][3]);
[U7Ø]  4Ø5:            }
[ØØØ]  4Ø6:
[743]  4Ø7:            if ((argc > 3) && (comp
[W83]  4Ø8:               (argv [3],"VAR") == Ø))
[P7Ø]  4Ø9:            {
[VW1]  41Ø:             var_flg = 1;
[R15]  411:             strcpy (par_buf,&argv[3][3]);
[U7Ø]  412:            }
[ØØØ]  413:
[943]  414:            if ((argc > 4) && (comp
[X83]  415:               (argv [4],"VAR") == Ø))
[P7Ø]  416:            {
[VW1]  417:             var_flg = 1;
[V15]  418:             strcpy (par_buf,&argv[4][3]);
[U7Ø]  419:            }
[ØØØ]  42Ø:
[2G2]  421:            if (var_flg == 1)
[P7Ø]  422:             {
[DY4]  423: /* Variante in PK uebernehmen */
[1T3]  424:              memset  (r_ptr+18,' ',15);
[3K3]  425: /* PK loeschen                 */
[5R4]  426:              strncpy (r_ptr+18, par_buf,
[313]  427:                       strlen (par_buf));
[U7Ø]  428:             }
[ØØØ]  429:
[345]  43Ø:             strncpy (m_ptr+23, r_ptr+18,1Ø)
        ;
[B45]  431: /* Variante von PK nach MODHDR */
[U7Ø]  432:           }
[ZRØ]  433:           else
[P7Ø]  434:           {
[L68]  435:            printf("UPK : Variante not found
        ... continue...\n");
[U7Ø]  436:           }
[ØØØ]  437:
[2H2]  438:           *(m_ptr+33) = '|';
[6F6]  439: /* Trennzeichen zum Versionseintrag */
[ØØØ]  44Ø:
[RZ5]  441: /*--------- Version uebernehmen --------*/
[ØØØ]  442:
[3X1]  443:           ver_flg = Ø;
[ØØØ]  444:
[RA5]  445:           if ((r_ptr = search(r_buf,r_buf+
[A35]  446:                n_read,"* Version       :"))
        != NULL)
[P7Ø]  447:           {
[8E6]  448:            if ((argc > 2) && (comp (argv [2
        ],"VER") == Ø))
[P7Ø]  449:            {
[7X1]  45Ø:             ver_flg = 1;
[8E4]  451:             strcpy (par_buf, argv[2]);
[U7Ø]  452:            }
[ØØØ]  453:
[BE6]  454:            if ((argc > 3) && (comp (argv [3
        ],"VER") == Ø))
[P7Ø]  455:            {
[7X1]  456:             ver_flg = 1;
[CE4]  457:             strcpy (par_buf, argv[3]);
[U7Ø]  458:            }
[ØØØ]  459:
[EE6]  46Ø:            if ((argc > 4) && (comp (argv [4
        ],"VER") == Ø))
[P7Ø]  461:            {
[7X1]  462:             ver_flg = 1;
[GE4]  463:             strcpy (par_buf, argv[4]);
[U7Ø]  464:            }
```

```
[ØØØ]  465:
[EG2]  466:           if (ver_flg == 1)
[P7Ø]  467:           {
[7M2]  468: /* VER gefunden */
[ØØØ]  469:
[AK4]  47Ø:           if ((i = strlen (par_buf)) > 3)
[P7Ø]  471:           {
[6A5]  472: /* VER hat noch eine Nummer dabei;
[V65]  473:    Versionsnummer uebernehmen     */
[1T3]  474:            memset  (r_ptr+18,' ',15);
[MB2]  475: /* PK loeschen */
[T96]  476:            strncpy (r_ptr+18, &par_buf[3],
        i - 3);
[U7Ø]  477:           }
[ZRØ]  478:           else
[P7Ø]  479:           {
[Ø65]  48Ø: /* VER steht alleine ohne Nummer;
[YS5]  481:    Versionsnummer + 1 auf ASCII-Basis */
[ØØØ]  482:
[ZK3]  483:            if ((*(r_ptr+2Ø) +=1) > '9')
[P7Ø]  484:            {
[ZE2]  485:             *(r_ptr+2Ø) = 'Ø';
[ØM3]  486:             if ((*(r_ptr+19) +=1) > '9')
[P7Ø]  487:             {
[SF2]  488:              *(r_ptr+19) = 'Ø';
[WL3]  489:              if ((*(r_ptr+18) +=1) > '9')
[P7Ø]  49Ø:              {
[NF2]  491:               *(r_ptr+18) = 'Ø';
[U7Ø]  492:              }
[U7Ø]  493:             }
[U7Ø]  494:            }
[U7Ø]  495:           }
[U7Ø]  496:          }
[VK4]  497: /* und nun die Versionsnummer
[B34]  498:    in den MODHDR kopieren      */
[245]  499:           strncpy (m_ptr+34, r_ptr+18,3);
[ØØØ]  5ØØ:
[2M3]  5Ø1: /* ver_buf vorbereiten
[4V3]  5Ø2:    fuer HOC-Bearbeitung */
[ØØØ]  5Ø3:
[N93]  5Ø4:           strcpy(ver_buf,"* ");
[8T4]  5Ø5:           strncat(ver_buf,r_ptr+18,3);
[G13]  5Ø6: /* "* nnn" erzeugen */
[U7Ø]  5Ø7:           }
[ØØØ]  5Ø8:
[FH7]  5Ø9:           else  error ("UPK : Version not
        found", NULL);
[ØØØ]  51Ø:
[ØØØ]  511:
[ØØ6]  512: /*--- History of Change bearbeiten ---*/
[ØØØ]  513:
[9V1]  514:           hoc_flg = Ø;
[ØØØ]  515:
[RA5]  516:           if ((r_ptr = search(r_buf,r_buf+
[S84]  517:                n_read,"/*HOC******")) != N
        ULL)
[P7Ø]  518:           {
[PT5]  519: /*  r_ptr steht am Anfang von HOC !! */
[AW6]  52Ø: /* "* nnn" -> Versionsnummer in HOC suchen
        */
[ØØØ]  521:
[LF9]  522:            if ((v_ptr = search(r_ptr,r_buf
        +n_read,ver_buf)) != NULL)
[P7Ø]  523:            {
[EH5]  524: /*---- Version in HOC gefunden -------*/
[Y95]  525:             strncpy (v_ptr+7, date_ptr ,8)
        ;
[3X1]  526:             ver_flg = Ø;
[NYA]  527:             printf("UPK : HOC-Service: Ver
        sion %s got a new date\n",ver_buf+2);
[AR5]  528: /*----- neues Datum in HOC-Feld -------*/
[U7Ø]  529:            }
[ZRØ]  53Ø:            else
[P7Ø]  531:            {
[C25]  532: /* Version in HOC nicht gefunden
[L94]  533:    -> Leerzeile suchen          */
[ØØØ]  534:
[5A4]  535:             if ((v_ptr = search(r_ptr,
[SA4]  536:                  r_buf+n_read,"*####")) !=
        NULL)
[P7Ø]  537:             {
[MM3]  538: /* "*####" in HOC gefunden */
[ØØØ]  539:
[394]  54Ø:              if ((h_ptr = search(v_ptr,
[SA4]  541:                   r_buf+n_read,"####*")) !
        = NULL)
[P7Ø]  542:              {
[MM3]  543: /* "####*" in HOC gefunden */
[ØØØ]  544:
[WR1]  545:               h_ptr += 6;
[SK2]  546: /* "####*" + CR + LF */
[114]  547:               n_line = h_ptr - v_ptr;
[PM5]  548: /* Anzahl zu kopierender Zeichen */
[H75]  549:               memcpy (comline,v_ptr,n_line)
        ;
```

```
[MM4]  55Ø: /* Einfuegezeile -> comline */
[ØØØ]  551:
[AH3]  552:                 strcat  (ver_buf,"  ");
[BX4]  553:                 memset  (comline+1,' ',n_line
       -3);
[PS3]  554: /* neue Zeile loeschen */
[KS3]  555:                 comline[n_line] = ØxØ;
[ØØØ]  556:
[687]  557:                 memcpy (comline,ver_buf,strle
       n (ver_buf));
[4G4]  558: /* neue Version in comline */
[B35]  559:                 memcpy (comline+7, date_ptr ,
       8);
[Ø34]  56Ø: /* neues Datum in comline */
[ØØØ]  561:
[7X1]  562:                 ver_flg = 1;
[8Z5]  563: /* fuer Editor, da neue Zeile in HOC  */
[ØØØ]  564:
[K83]  565:                 if ((mal_ptr = malloc
[UE3]  566:                     (BUFSIZE+512)) == NULL)
[ZM5]  567:                    error ("UPK : malloc faile
       d", NULL);
[ØØØ]  568:
[DV1]  569:                 hoc_flg = 1;
[8E2]  57Ø:                 r_ptr = r_buf;
[4V2]  571:                 h_ptr = mal_ptr;
[ØØØ]  572:
[EC3]  573:                 while (r_ptr < v_ptr)
[P7Ø]  574:                 {
[Ø35]  575: /*kopieren bis v_ptr erreicht */
[F33]  576:                  *h_ptr++ = *r_ptr++;
[U7Ø]  577:                 }
[ØØØ]  578:
[9N3]  579:                 for(i=Ø; i < n_line;i++)
[P7Ø]  58Ø:                 {
[PZ4]  581: /* neue HOC-Line einkopieren */
[GN3]  582:                  *h_ptr++ = comline[i];
[U7Ø]  583:                  }
[ØØØ]  584:
[LC2]  585:                 *h_ptr++ = ØxØA;
[AF1]  586:                 n_line++;
[ØØØ]  587:
[UM4]  588:                 while (r_ptr < r_buf+n_read)
[P7Ø]  589:                 {
[9H6]  59Ø: /*kopieren bis Ende von r_buf erreicht */
[F33]  591:                  *h_ptr++ = *r_ptr++;
[U7Ø]  592:                 }
[ØØØ]  593:
[PS2]  594:                 n_read += n_line;
[WP3]  595: /* n_read korrigieren */
[U7Ø]  596:                }
[ZRØ]  597:                else
[EH9]  598:                error ("UPK : End of Line »###
       #« in HOC not found", NULL);
[U7Ø]  599:               }
[ZRØ]  6ØØ:               else
[2S9]  6Ø1:               error ("UPK : Start of Line »*#
       ###« in HOC not found", NULL);
[U7Ø]  6Ø2:             }
[U7Ø]  6Ø3:           }
[ZRØ]  6Ø4:           else
[P7Ø]  6Ø5:           {
[NA6]  6Ø6:            printf("UPK : HOC-Service: not d
       one\n");
[3X1]  6Ø7:            ver_flg = Ø;
[U7Ø]  6Ø8:           }
[ØØØ]  6Ø9:
[5C4]  61Ø: /*-----------------------------------*/
[ØØØ]  611:
[6T4]  612:           strncpy (comline,m_ptr,47);
[2W2]  613:           comline[47] = ØxØ;
[2P3]  614:           printf ("%s\n",comline);
[WA4]  615: /* neuen MODHDR anzeigen */
[ØØØ]  616:
[8W1]  617:           hdr_flg = 1;
[7V2]  618: /* Kopf bearbeitet
[8A6]  619:    den Rest der Datei nur noch kopieren */
[U7Ø]  62Ø:         }
[1B1]  621:        r_ptr++;
[U7Ø]  622:       }
[ME2]  623:    if (hoc_flg == 1)
[P7Ø]  624:    {
[SU6]  625:    if ( write (fd2, mal_ptr, n_read) != n_
       read )
[4T7]  626:         error ("UPK :  write-error while co
       pying", NULL);
[ZRØ]  627:    else
[P7Ø]  628:         {
[ND2]  629:         free(mal_ptr);
[9V1]  63Ø:         hoc_flg = Ø;
[U7Ø]  631:         }
[ØØØ]  632:
[U7Ø]  633:    }
[ZRØ]  634:    else
[P7Ø]  635:    {
```

```
[1K6]  636:    if ( write (fd2, r_buf, n_read) != n_re
       ad )
[4T7]  637:         error ("UPK :  write-error whil
       e copying", NULL);
[U7Ø]  638:    }
[U7Ø]  639:   }
[ØØØ]  64Ø:
[N75]  641: /*---- Kopierschleife  E N D E -------*/
[ØØØ]  642:
[2A2]  643: if ( n_read < Ø)
[2P7]  644:      error ("UPK :  read-error while copyin
       g", NULL);
[5C4]  645: /*-----------------------------------*/
[3P1]  646: close (fd1);
[6P1]  647: close (fd2);
[ØØØ]  648:
[3D5]  649: /* Ueberpruefung, ob beide Dateien
[Z34]  65Ø:    gleich lang sind                 */
[ØØØ]  651:
[BK8]  652: if(filesize (newfile) != (filesize (argv[1]
       ) + n_line))
[P7Ø]  653:  {
[U3B]  654:   printf("\nUPK : files differ !!! check %s
       and %s !!! \n",newfile,argv[1]);
[PHF]  655:   printf("UPK : %s = %i    %s = %i \n",newf
       ile,filesize(newfile),argv[1],(filesize(arg
       v[1])+n_line));
[591]  656:   exit (1);
[U7Ø]  657:  }
[RU6]  658: /*else printf("\nUPK : updated file ok\n");
       */
[ØØØ]  659:
[8U3]  66Ø: strcpy(comline,"DEL ");
[C34]  661: strcat(comline,argv[1]);
[6U3]  662: /* Quelldatei loeschen */
[CW2]  663: system (comline);
[ØØØ]  664:
[CV3]  665: strcpy(comline,"REN ");
[B14]  666: strcat(comline,newfile);
[N44]  667: /* Zieldatei umbenennen */
[E23]  668: strcat(comline," ");
[C34]  669: strcat(comline,argv[1]);
[1M3]  67Ø: /* in alte Quelldatei   */
[CW2]  671: system (comline);
[ØØØ]  672:
[EG2]  673: if (ver_flg == 1)
[P7Ø]  674:    {
[Ø23]  675: /* Editor aufrufen,
[WU3]  676:    um HOC zu ergaenzen !!!! */
[614]  677:     strcpy(comline,editor);
[C34]  678:     strcat(comline,argv[1]);
[7N4]  679:     /* altes DIR merken, falls der
[KW3]  68Ø:        Editor das laufende DIR
[NB3]  681:        veraendert                  */
[P3B]  682:     printf ("\n\n\n\n=====================
       ============================\n");
[V2B]  683:     printf ("UPK : Neue Version -> Nachtrag
        von H O C notwendig\n\n\n");
[L28]  684:     printf ("UPK : Editing  >>> %s <<<\n\n\
       n",comline);
[ØD7]  685:     printf ("      Bitte eine Taste druecke
       n !!!!\n");
[EF9]  686:     printf ("=============================
       ====================\n");
[R61]  687:     getch();
[CW2]  688:     system (comline);
[U7Ø]  689:     }
[ØØØ]  69Ø:
[DA8]  691: chdir (act_dir);/* altes DIR wieder restaur
       ieren */
[ØØØ]  692:
[685]  693: /* Compiler-Parameter abfragen */
[ØØØ]  694:
[PF4]  695: if (com_flg == ØxFF) exit (1);
[ØØØ]  696:
[495]  697: if ((argc > 2) && (argv [2][Ø] == 'C'))
[P7Ø]  698:     {
[BW1]  699:     com_flg = 1;
[U7Ø]  7ØØ:     }
[ØØØ]  7Ø1:
[995]  7Ø2: if ((argc > 3) && (argv [3][Ø] == 'C'))
[P7Ø]  7Ø3:     {
[BW1]  7Ø4:     com_flg = 1;
[U7Ø]  7Ø5:     }
[ØØØ]  7Ø6:
[E95]  7Ø7: if ((argc > 4) && (argv [4][Ø] == 'C'))
[P7Ø]  7Ø8:     {
[BW1]  7Ø9:     com_flg = 1;
[U7Ø]  71Ø:     }
[ØØØ]  711:
[KF2]  712: if (com_flg == 1)
[P7Ø]  713:  {
[7K8]  714:  if ((r_ptr = search(compiler,compiler+6Ø,"
          )) != NULL)
```

Listing 1. »upk.c« nimmt Ihnen Verwaltungsarbeit bei der modularen Programmierung ab

```
[P7Ø]  715:      {
[FM2]  716:      *(r_ptr+2) = ØxØØ;
[U7Ø]  717:      }
[ØØØ]  718:
[4T7]  719:  if ((r_ptr = search(params,params+6Ø,"    "
       )) != NULL)
[P7Ø]  72Ø:      {
[272]  721:      *r_ptr = ØxØØ;
[U7Ø]  722:      }
[ØØØ]  723:
[LB4]  724:      strcpy(comline,compiler);
[ØØØ]  725:
[ND6]  726:      if ((r_ptr = strchr(comline,'+')) != NU
       LL)
[P7Ø]  727:           {
[3U1]  728:           *r_ptr = ' ';
[1N3]  729: /* "+" Zeichen loeschen */
[C34]  73Ø:           strcat(comline,argv[1]);
[6Ø7]  731: /* Compileraufruf mit Datei-Erweiterung */
[U7Ø]  732:           }
[945]  733:      else strcat(comline,wrkfile);
[E47]  734: /* Compileraufruf ohne Datei-Erweiterung */
[ØØØ]  735:
[NF2]  736:      if (par_flg == Ø)
[P7Ø]  737:           {
[G43]  738:           strcat(comline,"  ");
[DY3]  739:           strcat(comline,params);
[U7Ø]  74Ø:           }
[UE7]  741:      printf ("UPK : Compiling  >>> %s <<<\n"
       ,comline);
[CW2]  742:      system (comline);
[191]  743:      exit (Ø);
[U7Ø]  744:      }
[191]  745: exit (Ø);
[U7Ø]  746: }
```

Listing 1. »upk.c« nimmt Ihnen Verwaltungsarbeit bei der modularen Programmierung ab

```
[CT7]    1: /*****"MODHDR.: VERS.C       ZORLITE    |ØØ2
          21. 1.89"********
[G76]    2: ****************************************
          ******************
[8B6]    3: * Dateiname      : VERS.C
                             *
[YE6]    4: * Variante       : ZORLITE
                             *
[TR5]    5: * Version        : ØØ2
                             *
[AS5]    6: * Datum          : 21. 1.89
                             *
[P39]    7: * Autor(en)      : R.Manthey, Gasparistr.7,
          8ØØØ Muenchen 71 *
[G76]    8: ****************************************
          ******************
[YT5]    9: * Bedeutung:
                             *
[CN6]   1Ø: *-----------------------------------------
          -----------------*
[RE9]   11: * Eintrag des Link-Namens in LNKHDR mit Dat
          um Uhrzeit         *
[9K9]   12: * Eintrag der Vers.Nummer in RELHDR mit Dat
          um Uhrzeit         *
[269]   13: * Auflistung von Modul-, Link- und Release-
          Headerinfo         *
[XB6]   14: ****************************************
          ******************/
[L13]   15: #include <stdio.h>
[6A3]   16: #include <string.h>
[UR2]   17: #include <time.h>
[9L2]   18: #include <dos.h>
[ØØØ]   19:
[ØØØ]   2Ø:
[DS3]   21: #define  BUFSIZE   1Ø*1Ø24
[323]   22: /* Puffergroesse                */
[RH3]   23: #define  LEN_HDR       128
[TR4]   24: /* LNKHDR + RELHDR + Zuschlag*/
[YM3]   25: #define  LEN_MODH       45
[424]   26: /* Laenge von MODHDR           */
[MM3]   27: #define  LEN_LNKH       51
[D24]   28: /* Laenge von LNKHDR           */
[P83]   29: #define  PMODE           Ø
[KP3]   3Ø: #define  O_RDONLY        Ø
[EM3]   31: #define  NULL    (char *)Ø
[M13]   32: #define  ON              1
[R13]   33: #define  OFF             Ø
[ØØØ]   34:
[965]   35: /*------- Globale Variable ----------*/
[5Y2]   36: int       fd1,fd2,fd3;
[U85]   37: /* Dateidescriptoren  E / A          */
[FG2]   38: int       n_read ;
[KK5]   39: /* Anzahl gelesener Zeichen          */
[PZ3]   4Ø: int       b_read = BUFSIZE ;
[U96]   41: /* Anzahl der einzulesenden Zeichen  */
[XV2]   42: int       i,c              ;
[MR5]   43: /* Zaehlvariable fuer Schleife       */
[ØØØ]   44:
[TL2]   45: char      rel_flg;
[KU5]   46: /* Merkzelle fuer REL-Funktion       */
[8L2]   47: char      lnk_flg;
[1U5]   48: /* Merkzelle fuer LNK-Funktion       */
[YH2]   49: char      chk_flg;
[XZ5]   5Ø: /* Merkzelle fuer LNKHDR gefunden    */
[TK2]   51: char      div_flg;
[6X5]   52: /* Merkzelle fuer MODHDR geteilt     */
[ØØØ]   53:
[MA2]   54: char      *r_ptr;
[SF5]   55: /* Zeiger auf einzelne Zeichen       */
[5A2]   56: char      *c_ptr;
[726]   57: /* Zeiger auf einzelne Zeichen chksum*/
[FK2]   58: int       *ci_ptr;
[726]   59: /* Zeiger auf einzelne Zeichen chksum*/
[GW2]   6Ø: char      *date_ptr;
[S65]   61: /* Zeiger auf Zeitstring            */
[ØØØ]   62:
[W63]   63: char      hdrfile[2Ø];
[V96]   64: /* Dateiname fuer Zusammenstellung   */
[Z73]   65: char      newfile[2Ø];
[WT5]   66: /* Dateiname fuer neue Zieldatei     */
[MZ2]   67: char      linbuf[6Ø];
[3B6]   68: /* Zeilenpuffer fuer MODHDR LNKHDR   */
[573]   69: char      comline[4Ø];
[755]   7Ø: /* Commandline fuer DOS              */
[F84]   71: char      r_buf [BUFSIZE]  ;
[NS3]   72: /* Buffer                           */
[SD4]   73: /*..................................*/
[K52]   74: struct tm *bd;
[ØG1]   75: time_t t;
[5N5]   76: /*--- Globale Variable ---- E n d e ----*/
[ØØØ]   77:
[GL4]   78: /*---------------------------------------*/
[GN2]   79: int error (s1,s2)
[GL4]   8Ø: /*---------------------------------------*/
[M22]   81: char *s1 , *s2 ;
[P7Ø]   82: {
[H92]   83: printf (s1,s2);
[Z62]   84: printf ("\n");
[591]   85: exit (1);
[U7Ø]   86: }
[ØØØ]   87:
[GL4]   88: /*---------------------------------------*/
[W82]   89: int comp (m,s)
[GL4]   9Ø: /*---------------------------------------*/
[171]   91: char *m;
[RA5]   92: /* Zeiger auf einen Zeichenstring
[WF4]   93:    im Memory ohne \Ø                  */
[T71]   94: char *s;
[N95]   95: /* Zeiger auf einen Zeichen-Sting
[6Ø5]   96:    mit \Ø abgeschlossen               */
[P7Ø]   97: {
[5F3]   98:     for ( ; *s == *m; s++, m++)
[M92]   99:         if (*(s+1) == '\Ø')
[NS1]  1ØØ:             return (Ø);
[TE2]  1Ø1:         return (*s - *m);
[U7Ø]  1Ø2: }
[GL4]  1Ø3: /*---------------------------------------*/
[GN3]  1Ø4: char *search (ms,me,cs)
[NE6]  1Ø5: /* sucht in einem Memorybereich nach    */
[E85]  1Ø6: /* einem Zeichenstring                  */
[GL4]  1Ø7: /*---------------------------------------*/
[FB1]  1Ø8: char *ms;
[CH5]  1Ø9: /* Startpointer im Memory              */
[1B1]  11Ø: char *me;
[575]  111: /* Endpointer   im Memory              */
[7A1]  112: char *cs;
[9C6]  113: /* Referenzpointer auf Characterstring
[7S3]  114:    mit \Ø                              */
[P7Ø]  115: {
[NM3]  116: while (comp (ms,cs) != Ø)
[P7Ø]  117:     {
[AG4]  118:     if (++ms == me) return (NULL);
[4X2]  119: /* nicht gefunden */
[U7Ø]  12Ø:     }
[ØY1]  121: return (ms);
[H12]  122: /* gefunden */
[U7Ø]  123: }
[GL4]  124: /*---------------------------------------*/
[XA2]  125: char *date(ptm)
[GL4]  126: /*---------------------------------------*/
[2G2]  127: struct tm *ptm;
[P7Ø]  128: {
[677]  129:  static char d_line[2Ø] = "dd.mmm.19yy/hh:m
       m";
[5RB]  13Ø:  static char months[12*3+1] = "JanFebMrzApr
       MaiJunJulAugSepOktNovDez";
```

```
[ØØØ]  131:
[RE5]  132:  d_line[Ø] = (ptm->tm_mday >= 1Ø) ?
[574]  133:       (ptm->tm_mday / 1Ø) + 'Ø' : ' ';
[NP5]  134:  d_line[1] = (ptm->tm_mday % 1Ø) + 'Ø';
[ØØØ]  135:
[NP7]  136:  memcpy(&d_line[3],&months[ptm->tm_mon * 3]
       ,3);
[ØØØ]  137:
[AT5]  138:  d_line[9]  = (ptm->tm_year / 1Ø) + 'Ø';
[NV5]  139:  d_line[1Ø] = (ptm->tm_year % 1Ø) + 'Ø';
[ØØØ]  14Ø:
[HV5]  141:  d_line[12] = (ptm->tm_hour / 1Ø) + 'Ø';
[9V5]  142:  d_line[13] = (ptm->tm_hour % 1Ø) + 'Ø';
[ØØØ]  143:
[ØL5]  144:  d_line[15] = (ptm->tm_min / 1Ø) + 'Ø';
[UH5]  145:  d_line[16] = (ptm->tm_min % 1Ø) + 'Ø';
[ØØØ]  146:
[TD2]  147:  return d_line;
[U7Ø]  148: }
[GL4]  149: /*---------------------------------------*/
[B83]  15Ø: int main (argc,argv)
[GL4]  151: /*---------------------------------------*/
[532]  152: int       argc;
[RE2]  153: char *    argv[];
[ØØØ]  154:
[P7Ø]  155: {
[GL4]  156: /*---------------------------------------*/
[DD1]  157: time(&t);
[LK3]  158: /* get current time     */
[DZ2]  159: bd = localtime(&t);
[2B3]  16Ø: date_ptr = date (bd);
[ØØØ]  161:
[BD7]  162: printf ("VERS  V 1.1  LNKHDR- u. RELHDR-Inf
       o\n");
[FV6]  163: printf ("(C)RaMtY   Datum: %s\n",date_ptr);
[FX5]  164: printf ("-----------------------------------
       -\n");
[ØØØ]  165:
[ØØØ]  166:
[GL4]  167: /*---------------------------------------*/
[ØØØ]  168:
[VP1]  169: if (argc < 2)
[P7Ø]  17Ø:  {
[XW6]  171:  puts ("Anzahl der Argumente nicht ok !!!\n
       ");
[M3B]  172:  printf ("Aufruf :\nVERS  <Datei> {LNK{...1
       Ø...}} / {REL...23...}\n");
[591]  173:  exit (1);
[U7Ø]  174:  }
[ØØØ]  175:
[ØØØ]  176:
[GL4]  177: /*---------------------------------------*/
[ØØØ]  178:
[3N3]  179: /* Quelldatei oeffnen */
[UT2]  18Ø: strupr(argv[1]);
[H85]  181: /* Dateinamen in Grossbuchstaben
[C44]  182:    umwandeln                              */
[ØØØ]  183:
[TM6]  184: if (( fd1 = open   (argv[1], O_RDONLY)) ==
       -1)
[R26]  185:    error ("VERS :  can't open %s",argv[1]);
[ØØØ]  186:
[XM4]  187: if (argc > 2) strupr(argv[2]);
[AX4]  188: /* Steuerparameter in gross  */
[ØØØ]  189:
[ØØØ]  19Ø:
[TF4]  191: /* Dateinamen mit Extension
[8E3]  192:    ".HDR" erzeugen           */
[V34]  193: strcpy(hdrfile,argv[1]);
[ØØØ]  194:
[2G6]  195: if ((r_ptr = strchr(hdrfile,'.')) == NULL)
[CR3]  196:    strcat(hdrfile,".HDR");
[SE4]  197:   else strcpy(r_ptr,".HDR");
[ØØØ]  198:
[CZ3]  199: /* Header-Datei oeffnen */
[N76]  2ØØ: if (( fd2 = creat  (hdrfile, PMODE   )) ==
       -1)
[ZF6]  2Ø1:    error ("VERS :  can't create %s",hdrfile
       );
[ØØØ]  2Ø2:
[4W4]  2Ø3: /*.....................................*/
[ØØØ]  2Ø4:
[BW1]  2Ø5: if ((argc > 2)
[UL9]  2Ø6:  && ((comp (argv [2],"REL") == Ø) || (comp
       (argv [2],"LNK") == Ø)))
[P7Ø]  2Ø7:  {
[ØØØ]  2Ø8:
[GF4]  2Ø9:  if (comp (argv [2],"LNK") == Ø)
[P7Ø]  21Ø:  {
[L42]  211:  lnk_flg = ON;
[LU8]  212:  printf ("\nVERS : LNK-Eintrag  Datei: %s \
       n",argv [1]);
[U7Ø]  213:  }
[ØØØ]  214:
[5F4]  215:  if (comp (argv [2],"REL") == Ø)
[P7Ø]  216:  {
[842]  217:  rel_flg = ON;
[4V8]  218:  printf ("\nVERS : REL-Eintrag  Datei: %s \
       n",argv [1]);
[U7Ø]  219:  }
[ØØØ]  22Ø:
[TF4]  221: /* Dateinamen mit Extension
[4G3]  222:    ".VER" erzeugen           */
[K44]  223:    strcpy(newfile,argv[1]);
[ØØØ]  224:
[T9A]  225:  if ((r_ptr = strchr(newfile,'.')) == NULL)
       strcat(newfile,".VER");
[RF4]  226:   else strcpy(r_ptr,".VER");
[ØØØ]  227:
[ZH3]  228:  /* Zieldatei oeffnen */
[U86]  229:  if (( fd3 = creat  (newfile, PMODE   )) ==
       -1)
[UG6]  23Ø:         error ("VERS :  can't create %s",n
       ewfile);
[ØØØ]  231:
[U7Ø]  232:  }
[ZRØ]  233: else
[P7Ø]  234:  {
[UY8]  235:  printf ("\nVERS : DIR-Funktion  Datei %s \
       n",argv [1]);
[7A2]  236:  rel_flg = OFF;
[KA2]  237:  lnk_flg = OFF;
[U7Ø]  238:  }
[ØØØ]  239:
[8R2]  24Ø: /* --- 1. Durchlauf:
[235]  241:    Lage von LNKHDR ermitteln --- */
[ØØØ]  242:
[E92]  243: chk_flg = OFF;
[ØØØ]  244:
[RL8]  245: printf ("VERS : ... reading %s for check ..
       .\n",argv[1]);
[Y67]  246: while (( n_read = read (fd1,r_buf,BUFSIZE))
       > Ø)
[P7Ø]  247:  {
[MK9]  248:  if ((r_ptr = search(r_buf,r_buf+n_read,"LN
       KHDR.:")) != NULL)
[P7Ø]  249:     {
[8C4]  25Ø: /* ..... LNKHDR gefunden .... */
[DV5]  251:     if (r_ptr+(LEN_HDR) > r_buf+n_read)
[PL4]  252:        b_read = BUFSIZE-(LEN_HDR);
[F32]  253:     chk_flg = ON;
[PØ1]  254:     break;
[U7Ø]  255:     }
[U7Ø]  256:  }
[3P1]  257: close (fd1);
[ØØØ]  258:
[3X2]  259: if (chk_flg == OFF)
[947]  26Ø:    error("VERS : LNKHDR not found ... bye",
       NULL);
[ØØØ]  261:
[9R2]  262: /* --- 2. Durchlauf:
[XB5]  263:    LNKHDR und RELHDR bearbeiten --- */
[ØØØ]  264:
[Y45]  265: fd1 = open (argv[1], O_RDONLY);
[GR3]  266: /* hier ohne Pruefung */
[ZN9]  267: printf ("VERS : ... proceeding %s with %i\n
       ",argv[1],b_read);
[ØØØ]  268:
[ØØØ]  269:
[HØ7]  27Ø: while (( n_read = read (fd1,r_buf,b_read))
       > Ø)
[P7Ø]  271:  {
[MK9]  272:  if ((r_ptr = search(r_buf,r_buf+n_read,"LN
       KHDR.:")) != NULL)
[P7Ø]  273:  {
[8C4]  274: /* ..... LNKHDR gefunden .... */
[WP2]  275:   if (lnk_flg == ON)
[D64]  276: /* LNK-Eintrag vornehmen */
[P7Ø]  277:   {
[ØM3]  278:    memset  (r_ptr+9,' ',25);
[6M7]  279:    strncpy (r_ptr+9, argv [1],strlen (argv
       [1]));
[6U8]  28Ø:    strncpy (r_ptr+22, (argv [2])+3,(strlen
       (argv [2])-3));
[2M3]  281:    memset  (r_ptr+32,' ',2);
[CE5]  282:    strncpy (r_ptr+34, date_ptr ,17);
[U7Ø]  283:   }
[ØØØ]  284:
[FS2]  285:   linbuf [Ø] = ØxØC;
[162]  286: /* Form-Feed  */
[LS2]  287:   linbuf [1] = ØxØD;
[BXØ]  288: /* CR  */
[FS2]  289:                 linbuf [2] = ØxØA;
[WWØ]  29Ø: /* LF  */
[P16]  291:   strncpy (linbuf+3,r_ptr,LEN_LNKH);
[5E4]  292:   linbuf [LEN_LNKH+3] = ØxØD;
[TP2]  293: /* CR anhaengen  */
[ZD4]  294:   linbuf [LEN_LNKH+4] = ØxØA;
```

Listing 2. »vers.c« verwaltet Link-, Release- und Modul-Informationen

```
[AP2]  295: /* LF anhaengen  */
[FC4]  296:   linbuf [LEN_LNKH+5] = Øx ØØ;
[EN2]  297: /* \Ø anhaengen  */
[ØØØ]  298:
[PE3]  299:   printf("  %s",linbuf+3);
[263]  3ØØ: /* LNKHDR anzeigen  */
[VY4]  3Ø1:   write (fd2, linbuf, LEN_LNKH+5);
[EX4]  3Ø2: /* und in List-Datei schreiben */
[ØØØ]  3Ø3:
[6F3]  3Ø4: /* bei LNKHDR-Funktion
[R94]  3Ø5:    RELHDR-Eintrag loeschen */
[ØØØ]  3Ø6:
[P16]  3Ø7: /* RELHDR muss  n a c h LNKHDR stehen ! */
[ØØØ]  3Ø8:
[6D5]  3Ø9:    if ((r_ptr = search(r_ptr,r_buf+
[A94]  31Ø:         n_read,"RELHDR.:")) == NULL)
[U47]  311:       error("VERS : RELHDR not found ... by
       e",NULL);
[ØØØ]  312:
[GW3]  313: /* ... RELHDR gefunden ... */
[WP2]  314:    if (lnk_flg == ON)
[P7Ø]  315:    {
[P23]  316: /* RELHDR loeschen */
[UL3]  317:     memset  (r_ptr+9,' ',42);
[ØØØ]  318:
[TY3]  319: /* Checksumme = Øx4Ø4Ø = '@@'
[MP4]  32Ø:    wenn keine generiert wurde */
[454]  321:     ci_ptr = (int*)(r_ptr + 32);
[4G2]  322:     *ci_ptr = Øx4Ø4Ø;
[U7Ø]  323:    }
[ØØØ]  324:
[HP2]  325:    if (rel_flg == ON)
[464]  326: /* REL-Eintrag vornehmen */
[P7Ø]  327:   {
[ØM3]  328:    memset  (r_ptr+9,' ',25);
[4M8]  329:    strncpy (r_ptr+9, (argv [2])+3,(strlen (
       argv [2])-3));
[2M3]  33Ø:    memset  (r_ptr+32,' ',2);
[CE5]  331:    strncpy (r_ptr+34, date_ptr ,17);
[ØØØ]  332:
[H95]  333: /*---- Ckecksumme generieren ------*/
[BSØ]  334:    c = Ø;
[8R2]  335:    c_ptr = r_ptr+9;
[ØØØ]  336:
[ED5]  337:    for (i=Ø; i < 23; i++) c += *(c_ptr++);
[ØØØ]  338:
[454]  339:    ci_ptr = (int*)(r_ptr + 32);
[EM2]  34Ø:    *ci_ptr = ~c+1;
[SR2]  341: /* Komplement + 1 */
[814]  342: /*---------------------------------*/
[U7Ø]  343:    }
[ØØØ]  344:
[NS5]  345:    strncpy (linbuf,r_ptr,LEN_LNKH);
[W74]  346:    linbuf [LEN_LNKH]   = ØxØD;
[TP2]  347: /* CR anhaengen  */
[PD4]  348:    linbuf [LEN_LNKH+1] = ØxØA;
[AP2]  349: /* LF anhaengen  */
[6C4]  35Ø:    linbuf [LEN_LNKH+2] = Øx ØØ;
[EN2]  351: /* \Ø anhaengen  */
[ØØØ]  352:
[B83]  353:    printf("  %s",linbuf);
[T53]  354: /* RELHDR anzeigen  */
[KY4]  355:    write (fd2, linbuf, LEN_LNKH+2);
[EX4]  356: /* und in List-Datei schreiben */
[U7Ø]  357:   }
[ØØØ]  358:
[S46]  359:   if ((rel_flg == ON) || (lnk_flg == ON))
[P7Ø]  36Ø:    {
[5K6]  361:     if ( write (fd3, r_buf, n_read) != n_re
       ad )
[328]  362:      error ("VERS :  write-error while copyi
       ng", NULL);
[U7Ø]  363:    }
[U7Ø]  364:     }
[ØØØ]  365:
[3P1]  366: close (fd1);
[ØØØ]  367:
[S46]  368: if ((rel_flg == ON) || (lnk_flg == ON))
[9P1]  369:     close (fd3);
[ØØØ]  37Ø:
[AR2]  371: /* --- 3. Durchlauf:
[TH3]  372:    MODHDR auflisten --- */
[ØØØ]  373:
[PH5]  374: /* Quelldatei noch einmal oeffnen */
[TM6]  375: if (( fd1 = open   (argv[1], O_RDONLY)) ==
       -1)
[SZ7]  376:      error ("VERS :  can't open %s for MODH
       DR",argv[1]);
[ØØØ]  377:
[RSØ]  378: i = Ø;
[BF6]  379: /* falls MODHDR geteilt, enthaelt i die
[4B6]  38Ø:    Anzahl der bereits gelesenen Zeichen */
[ØØØ]  381:
[Y67]  382: while (( n_read = read (fd1,r_buf,BUFSIZE))
        > Ø)
```

```
[P7Ø]  383:  {
[8E2]  384:   r_ptr = r_buf;
[NZ4]  385:   while (r_ptr < (r_buf + n_read))
[P7Ø]  386:    {
[L91]  387:     if (i != Ø)
[TW3]  388: /* Linbuf ergaenzen ??? */
[P7Ø]  389:     {
[V25]  39Ø: /* Rest von linbuf auffuellen */
[US6]  391:      strncpy (linbuf+i,r_ptr,(LEN_MODH-1-i)
       );
[4S3]  392:      r_ptr += (LEN_MODH-1-i);
[2A3]  393:      printf("++%s",linbuf);
[G53]  394: /* MODHDR anzeigen  */
[VX4]  395:      write (fd2, linbuf, LEN_MODH+2);
[EX4]  396: /* und in List-Datei schreiben */
[E13]  397:      r_ptr += LEN_MODH;
[GX5]  398: /* kopierten MODHDR ueberspringen */
[RSØ]  399:      i = Ø;
[U7Ø]  4ØØ:     }
[ØØØ]  4Ø1:
[6D5]  4Ø2:     if ((r_ptr = search(r_ptr,r_buf+
[674]  4Ø3:          n_read,"MODHDR.:")) != NULL)
[P7Ø]  4Ø4:     {
[ZB4]  4Ø5: /* ..... MODHDR gefunden .... */
[ØØØ]  4Ø6:
[YH5]  4Ø7:      if(r_ptr+LEN_MODH > r_buf+n_read)
[P7Ø]  4Ø8:      {
[574]  4Ø9: /* MODHDR liegt auf Grenze
[525]  41Ø:    -> Rest nach linbuf kopieren */
[A84]  411:       i = (r_buf+n_read)- r_ptr;
[ZE4]  412:       strncpy (linbuf,r_ptr,i);
[C44]  413:       r_ptr = (r_buf + n_read);
[ØP3]  414: /* Schleifenausstieg */
[U7Ø]  415:      }
[ZRØ]  416:      else
[P7Ø]  417:      {
[1P6]  418: /* MODHDR kann komplett kopiert werden */
[CS5]  419:       strncpy (linbuf,r_ptr,LEN_MODH);
[574]  42Ø:       linbuf [LEN_MODH]   = ØxØD;
[TP2]  421: /* CR anhaengen  */
[ZC4]  422:       linbuf [LEN_MODH+1] = ØxØA;
[AP2]  423: /* LF anhaengen  */
[FB4]  424:       linbuf [LEN_MODH+2] = Øx ØØ;
[EN2]  425: /* \Ø anhaengen  */
[B83]  426:       printf("  %s",linbuf);
[G53]  427: /* MODHDR anzeigen  */
[VX4]  428:       write (fd2, linbuf, LEN_MODH+2);
[EX4]  429: /* und in List-Datei schreiben */
[E13]  43Ø:       r_ptr += LEN_MODH;
[GX5]  431: /* kopierten MODHDR ueberspringen */
[U7Ø]  432:      }
[U7Ø]  433:     }
[ZRØ]  434:     else
[C44]  435:      r_ptr = (r_buf + n_read);
[ØP3]  436: /* Schleifenausstieg */
[U7Ø]  437:    }
[U7Ø]  438:   }
[3P1]  439: close (fd1);
[6P1]  44Ø: close (fd2);
[ØØØ]  441:
[7E5]  442: /*---------- Abschluss -----------------*/
[ØØØ]  443:
[S46]  444: if ((rel_flg == ON) || (lnk_flg == ON))
[P7Ø]  445:     {
[3D5]  446: /* Ueberpruefung, ob beide Dateien
[Z34]  447:    gleich lang sind                 */
[ØØØ]  448:
[DV6]  449:     if(filesize (argv[1]) != filesize (newf
       ile))
[P7Ø]  45Ø:      {
[5DB]  451:       printf("\nVERS : files differ !!! chec
       k %s and %s !!! \n",newfile,argv[1]);
[591]  452:       exit (1);
[U7Ø]  453:      }
[ØØØ]  454:
[8U3]  455:      strcpy(comline,"DEL ");
[C34]  456:      strcat(comline,argv[1]);
[6U3]  457: /* Quelldatei loeschen */
[CW2]  458:      system (comline);
[ØØØ]  459:
[CV3]  46Ø:      strcpy(comline,"REN ");
[B14]  461:      strcat(comline,newfile);
[N44]  462: /* Zieldatei umbenennen */
[E23]  463:      strcat(comline," ");
[C34]  464:      strcat(comline,argv[1]);
[1M3]  465: /* in alte Quelldatei   */
[CW2]  466:      system (comline);
[ØØØ]  467:
[U7Ø]  468:     }
[ØØØ]  469:
[191]  47Ø: exit (Ø);
[U7Ø]  471: }
```

Listing 2. »vers.c« verwaltet Link-, Release- und Modul-Informationen

Virus im Visier

Haben Sie Probleme mit »trojanischer Software«? Wollen Sie in Zukunft solche Probleme vermeiden? Das in C geschriebene Programm »Vircheck« hilft Ihnen, Ihre Festplatte schnell und sicher vor der Virenplage zu schützen.

Viren, die sich an den Anfang oder das Ende eines ausführbaren Programms kopieren, sind heimtückisch. Von harmlosen piepsenden Nervtötern, aber auch von Festplattenkillern ist oft die Rede, wenn es um Viren geht. Meistens wird nur Panik gemacht, aber selten erklärt, wie man sich vor Viren schützt.

In der DOS International (Ausgabe 12'88) wurde ein Ansatz zum Aufspüren der gefährlichen Tierchen aus Troja bereits beschrieben (1). In diesem Beispiel beschränkte sich der Autor darauf, lediglich die Dateigröße und die Dateiattribute zu speichern und auf deren Veränderung hin zu überprüfen.

Andere Viruserkennungsprogramme basieren üblicherweise auf dem Errechnen einer Prüfsumme. Diese Prüfsumme speichert das Programm und überprüft beim nächsten Durchlauf, ob sich Veränderungen ergeben haben. Diese Programme kosten häufig viel Zeit. Das liegt daran, daß die Checksumme über die gesamte Datei berechnet wird.

Sie werden sich sicherlich denken können, wie lange Sie auf die Virusüberprüfung warten müssen, insbesondere wenn Ihre Festplatte sehr voll ist. Sinn hat ein Viruserkennungsprogramm jedoch nur dann, wenn es möglichst oft, am besten täglich, gestartet wird. Nur so haben Sie die Chance, einem Virus rechtzeitig auf die Spur zu kommen.

Das abgedruckte Programm »vircheck.c« spart Bearbeitungszeit. Die Idee, die dem Programm zugrunde liegt, basiert auf dem grundsätzlichen Aufbau einer Datei, die ein ausführbares Programm enthält. In der DOS International 3'88 (2) wurde ausführlich auf den Aufbau einer Exe- beziehungsweise Com-Datei eingegangen. Dieser Beitrag zeigt, daß fast alle wichtigen Informationen, die zum Starten eines Programms notwendig sind, in den ersten 128 Byte der Programmdatei stehen, sei es das PSP (Program Segment Prefix), das MS-DOS vor jede Com-Datei schreibt, oder der Exe-Header vor jeder Exe-Datei.

Nur für Viren relevante Daten werden überprüft, um »vircheck.c« zu beschleunigen

Das heißt, eine Checksummenberechnung ist nur über die ersten 128 Byte notwendig. Bei einer Programmänderung (eventuell durch ein Virusprogramm) müssen nämlich auch diese Header-Daten geändert werden. Diese Kenntnis macht sich das Viruserkennungsprogramm »vircheck.c« zunutze.

Virusprüfsoftware verlangt häufig als Eingabe eine Liste der Programmdateien, die sie prüfen soll. Diese Liste muß der Anwender erst mühsam zusammenstellen. Wie häufig passiert es dabei, daß man eine Datei vergißt oder sich vertippt?

Das hier abgedruckte Programm verlangt als Eingabe keine Dateinamen, sondern nur eine Liste von Inhaltsverzeichnissen. Es durchsucht automatisch alle Com- und Exe-Dateien in den angegebenen Verzeichnissen.

»vircheck.c« ist in Turbo C geschrieben. Damit Sie diesen nützlichen Helfer für alle Turbo-C-Compiler (Version 1.0 und Version 1.5) nutzen können, wurde eine Header-Datei »grafik.h« mit dem Include-Befehl integriert. Diese Header-Datei realisiert durch direkte BIOS-Aufrufe schnelle Textausgabefunktionen auf dem Bildschirm.

```
c:\
c:\bin\dos
c:\utl
```

Bild 1. So kann eine Eingabedatei aussehen; sie enthält nur Verzeichnisnamen, keine Dateinamen

Das Hauptprogramm, das die Routinen von »grafik.h« aufruft, unterteilt sich in verschiedene Funktionen. Die Tabelle zeigt das Zusammenspiel dieser Funktionen, die teilweise rekursiv arbeiten. Eine dieser Routinen ist »bedir()«, die eine Eingabedatei untersucht.

Das Programm verlangt diese Eingabedateien mit den Inhaltsverzeichnissen, die es bearbeiten soll. Die Eingabedatei (Beispiel in Bild 1) schreiben Sie mit einem beliebigen Editor, der ASCII-Dateien erzeugt (Edlin, Word, Norton Editor). Vircheck durchsucht mit der Routine »bedir()« die angegebenen Verzeichnisse automatisch und berechnet die Prüfsummen der dort vorhandenen Exe-, Sys-, Obj- und Com-Dateien. Diese Checksummen werden in einer List-Datei »vircheck.lst« abgelegt (Bild 2).

Falls eine Datei bereits bearbeitet wurde, vergleicht der Viruschecker die alte Prüfsumme mit der neu berechneten. Stimmen diese Summen nicht überein, erscheint eine Virus-Warnung auf dem Bildschirm. Das Programm wartet dann einige Sekunden (Sleep-Statement) und setzt die Bearbeitung fort. Die Warnungen dokumentiert es in der Datei »virus.lst« (Bild 3). Falls die Datei noch nicht bearbeitet wurde, fügt sie der Virusprüfer automatisch der Prüfsummenliste an.

Über Parameter (Bild 4) legen Sie fest, ob eine Bildschirmausgabe erfolgt oder nicht. Eine Ausgabe der Checksummenberechnungen macht die Bearbeitung etwa sechsmal langsamer. Falls Sie keine Bildschirmausgabe wünschen, gibt Vircheck für je-

```
c:\IO.SYS    1628
c:\MSDOS.SYS    1524
c:\COMMAND.COM    2d52
c:\CONFIG.SYS    281a
c:\VIRCHECK.EXE    72c
c:\bin\dos\ASSIGN.COM    c06
c:\bin\dos\ATTRIB.EXE    885
c:\bin\dos\BACKUP.EXE    899
c:\bin\dos\CHKDSK.EXE    714
c:\bin\dos\COMMAND.COM    2d52
c:\bin\dos\COMP.COM    10d4
c:\bin\dos\DEBUG.EXE    80b
c:\bin\dos\DISKCOMP.EXE    575
```

Bild 2. Die Datei »vircheck.lst« wird vom Virusprüfer erzeugt, sie enthält die geprüften Dateinamen mit deren Prüfsumme

```
Virus Checker V 1.0 (c) by Reklies.  Unstimmigkeit in Liste!
Virusgefahr!! Datei : c:\VIRCHECK.EXE    72c
neue Checksumme:      ======>>>> 0x7de
```

Bild 3. Bei Checksummen-Fehler erfolgt eine Warnung auf dem Bildschirm, die in der Datei »virus.lst« dokumentiert wird

Funktion:	Aufgabe:
unsigned chksum()	Berechnung der Checksumme
void pruef()	Überprüfung der neu errechneten Checksumme
void arbeite()	Abarbeitung eines Verzeichnisses / Bearbeitung einer Datei (rekursive Funktion)
void bedir()	Vorbereitungsarbeiten für die Funktion »arbeite()« / Bestimmen, welche Dateien bearbeitet werden sollen
main(argc, argv)	Main-Funktion von »vircheck.c«

Tabelle. Die Funktionen des Programms »vircheck.c«

```
        Virus CHECKER  V 1.0     (c) 1988 by Reklies

Fehlerhafte Bedienung!

Korrekte Benutzung: virus dateiname parameter
            oder: virus pfad\dateiname parameter

Die Datei muß eine Liste von Directories enthalten,
die bearbeitet werden sollen.
Pro Zeile ist ein Directoryeintrag zulässig.
Die zulässig maximale Pfadlänge ist 68 Bytes.

Parameter /s => Bildschirmausgabe unterdrücken.
Parameter /n => Checkliste wird neu erstellt.
```

Bild 4. Diese Meldungen erscheinen, wenn Sie »Vircheck« ohne Parameter aufrufen

des bearbeitete Verzeichnis einen Punkt aus. Mit einem weiteren Parameter bestimmen Sie, ob eine komplette neue Checkliste (»vircheck.lst«) ohne Prüfung der Checksummen angelegt wird. Das ist bei der erstmaligen Benutzung oder bei häufigen Programmänderungen teilweise notwendig.

Zum Schluß noch einige Leistungsbeweise, die auf einem 10-MHz-AT mit einem Waitstate und einer 40-MByte-Festplatte (durchschnittliche Zugriffszeit: 30 Millisekunden) gemessen wurden. Compiliert wurde das Programm mit Turbo C 1.5, für die Übersetzung wurde der 80286-Befehlssatz und »Optimization = Speed« gewählt.

Das Erzeugen einer neuen Checkliste (ohne Prüfen der Summen und ohne Bildschirmausgabe) benötigt bei 175 Verzeichnissen mit 586 relevanten Dateien nur 43 Sekunden. Das Erzeugen und Überprüfen von »vircheck.lst« dauert bei zwei Verzeichnissen und 48 Dateien nur zirka fünf Sekunden. Zur Bearbeitung von 175 Verzeichnissen mit 586 zu prüfenden Dateien braucht das Programm vier Minuten und 42 Sekunden. Für den durchschnittlichen MS-DOS-Anwender mit wesentlich weniger Verzeichnissen und Dateien ist eine Integration des Virusprüfers in die Datei »autoexec.bat« folglich kein wesentlicher Zeitverlust. Der Gewinn, nämlich höhere Sicherheit, überwiegt bei weitem den Verlust.

Im Programmlisting finden Sie auch Präprozessor-Anweisungen. Der Code, der im Programmlisting zwischen »#ifdef TRACE« und »#endif TRACE« steht, ist für den Programmablauf nicht relevant. Für C-Programmierer heißt das, daß bei Programmänderungen diese Anweisungen sinnvoll sind, weil damit ein »Trace Output« in einer »Trace-Datei« erzeugt wird (Compilerangabe »DEFINE = TRACE«). Mit Hilfe dieser Anweisung wird die Arbeit von Vircheck mitprotokolliert.

Wer das Programm erweitern will, dem bieten sich viele Optionen. So können Sie zum Beispiel Dateistatus-Informationen zusätzlich in die Checkliste aufnehmen und mitprüfen. Außerdem kommt das Programm bei entsprechender Änderung auch ohne Angabe der Verzeichnisnamen aus: Der Anwender muß dann nur das zu überprüfende Laufwerk angeben. Den eigenen Ideen sind hier keine Grenzen gesetzt.

(Bernd Reklies/mk)

Weiterführende Literatur:
(1) »Computerviren schnell erkannt«, DOS International 12'88
(2) »Die Unterschiede zwischen Com- und Exe-Dateien«, DOS International 3'88
(3) Ralf Burger, »Das große Computer-Viren Buch, 1987«, Düsseldorf, Data Becker Verlag

```
[CF4]    1: /*****************************************
[CC6]    2: *  Programm:  Virus Check Programm        *
[9D4]    3: *  Version:   1.Ø                         *
[W85]    4: *  Dateiname: vircheck.c                  *
[6C6]    5: *  Zweck:     Dieses Programm berechnet   *
[1X5]    6: *             die Checksumme aller Dateien*
[7V5]    7: *             in den angegebenen Inhalts- *
[4H6]    8: *             verzeichnissen. Es überprüft*
[TT5]    9: *             den Wert mit einer eingetra-*
[RS5]   1Ø: *             genen Liste und gibt evtl.  *
[FP5]   11: *             Virusalarm bei Verdacht     *
[XA6]   12: *  Aufruf:    Der Aufruf erfolgt durch    *
[FL5]   13: *             vircheck datei <return>     *
[9N5]   14: *  Compiler:  Turbo C V1.Ø / V1.5          *
[WE3]   15: *                                         *
[MD6]   16: *  Autor: Dipl.-Inform. Bernd Reklies      *
[PF4]   17: *****************************************/
[L13]   18: #include <stdio.h>
[1L2]   19: #include <dir.h>
[VG2]   2Ø: #include <io.h>
[RØ3]   21: #include <fcntl.h>
[2Ø3]   22: #include <conio.h>
[413]   23: #include <errno.h>
[AØ3]   24: #include <ctype.h>
[VØ3]   25: #include "grafik.h"
[ØØØ]   26:
[EU3]   27: #define BOOL     unsigned
[6L2]   28: #define TRUE      1
[8K2]   29: #define FALSE     Ø
[PD2]   3Ø: #define AND       &&
[WR2]   31: #define OR        ||
[RC2]   32: #define NOT       !
[AD2]   33: #define EQ        ==
[4H2]   34: #define NEQ       !=
[ØØØ]   35:
[E36]   36: #define MAX_BYTES  128  /* Anzahl Bytes
```

```
[Ø13]   37:                          in der Checksumme */
[ØØØ]   38:
[KE3]   39:                        /* Globale Variable : */
[N16]   4Ø: struct  ffblk ffblock;  /* File block */
[CR3]   41: char drive[MAXDRIVE];
[T83]   42: char dir  [MAXDIR];
[UE3]   43: char file [MAXFILE];
[DE3]   44: char suff [MAXEXT];
[ØØØ]   45:
[8W5]   46: unsigned char buffer[MAX_BYTES+1];
[KK1]   47: /* Buffer */
[1W4]   48: char     autor[]   = "Reklies\Ø";
[AW3]   49: unsigned anz_files = Ø;
[MU3]   5Ø: BOOL     virus     = FALSE;
[R75]   51:          /* Unstimmigkeit in Checkdatei */
[854]   52: BOOL     neueliste = FALSE;
[EØ5]   53:            /* Neue Checkliste erstellen? */
[KR3]   54: BOOL     ausgabe   = TRUE;
[7L3]   55:            /* Bildschirmausgabe */
[ZL3]   56: FILE     *fopen(),*fpout;
[ØØØ]   57:
[5V1]   58: #ifdef TRACE
[GE2]   59: FILE  *fptrace;
[NZØ]   6Ø: #endif
[ØØØ]   61:
[ØG4]   62: /*****************************************/
[1S4]   63: /* Berechung der Checksumme */
[ØØØ]   64:
[DU3]   65: unsigned chksum(filed)
[ZW1]   66: int   filed;
[P7Ø]   67: {
[RB2]   68: int       anzahl,
[BVØ]   69:           lauf;
[PY2]   7Ø: unsigned summe=Ø;
[ØØØ]   71:
[RB6]   72: anzahl=_read(filed,buffer,MAX_BYTES);
```

```
[355]   73: for (lauf=Ø;lauf<anzahl;lauf++)
[P7Ø]   74:   {
[RR3]   75:   summe += buffer[lauf];
[5X1]   76:   if (ausgabe)
[P7Ø]   77:     {
[7P1]   78:     move(8,4Ø);
[UC3]   79:     printf("Øx%4x",summe);
[U7Ø]   8Ø:     }
[U7Ø]   81:   }
[6D2]   82: return(summe);
[U7Ø]   83: }
[GS3]   84:                     /* end function chksum */
[ØØØ]   85:
[ØG4]   86: /*****************************************/
[4D4]   87:                   /* function pruef start: */
[ØØØ]   88:
[CH3]   89: void pruef(fname,sum)
[7H2]   9Ø: char     fname[];
[9A2]   91: unsigned sum;
[P7Ø]   92: {
[3K1]   93: int hilf;
[XF2]   94: unsigned wert;
[MT2]   95: char string[81];
[1A2]   96: char name[7Ø];
[BX2]   97: BOOL found=FALSE;
[E92]   98: char *hilfptr;
[CØ2]   99: FILE *fpvir;
[ØØØ]  1ØØ:
[PN6]  1Ø1: fpout=freopen("vircheck.lst","r",fpout);
[3V2]  1Ø2: if (fpout EQ NULL)
[P7Ø]  1Ø3:   {
[E2C]  1Ø4:     printf ("\n\n\a\apruef: VIRCHECK.LST lä
          ßt sich nicht öffnen!\n\n");
[LF1]  1Ø5:     cur_on();
[F61]  1Ø6:     exit(2);
[U7Ø]  1Ø7:   }
[NBØ]  1Ø8: do
[P7Ø]  1Ø9:   {
[FC5]  11Ø:   hilfptr=fgets(string,8Ø,fpout);
[393]  111:   if (hilfptr EQ NULL)
[P7Ø]  112:     {
[PØ1]  113:     break;
[U7Ø]  114:     }
[ØØØ]  115:
[9Y1]  116: #ifdef TRACE2
[XFB]  117: fprintf(fptrace,"pruef:fgets string= %3Øs e
          rrno %d\n",string,errno);
[5Z1]  118: #endif TRACE2
[ØØØ]  119:
[MA7]  12Ø: if (strncmp(string,fname,strlen(fname)) EQ
          Ø)
[P7Ø]  121:     {
[132]  122:     found=TRUE;
[U7Ø]  123:     }
[U7Ø]  124:   }
[AU2]  125: while (NOT found);
[ØØØ]  126:
[AH1]  127: if (found)
[P7Ø]  128:   {
[G66]  129:   hilf=sscanf(string,"%s%x",name,&wert);
[ØØØ]  13Ø:
[9Y1]  131: #ifdef TRACE2
[D2C]  132: fprintf(fptrace,"pruef: sscanf: return=%d n
          ame= %s lwert=%x errno %d\n",\
[WP3]  133:       hilf,name,wert,errno);
[5Z1]  134: #endif TRACE2
[ØØØ]  135:
[TM1]  136:  if (hilf<Ø)
[P7Ø]  137:   {
[MY6]  138:   printf ("\n\n\a\apruef: Error in sscanf !
[GN5]  139:         ret=%d errno= %d!\n\n",\hilf,errno);
[LF1]  14Ø:   cur_on();
[8L2]  141:   fflush(stdout);
[F61]  142:   exit(2);
[U7Ø]  143:   }
[ØØØ]  144:
[ØR2]  145:   if (wert NEQ sum)
[1N4]  146:                 /* keine Übereinstimmung!! */
[P7Ø]  147:     {
[B72]  148:     virus = TRUE;
[8N4]  149:               /* Checksumme stimmt nicht! */
[ØØØ]  15Ø:
[NM4]  151:     fpvir=fopen("virus.lst","a");
[FV2]  152:     if (fpvir EQ NULL)
[P7Ø]  153:       {
[2CB]  154:       printf ("\n\n\a\apruef: Virus.lst läß
          t sich nicht öffnen!\n\n");
[LF1]  155:       cur_on();
[F61]  156:       exit(2);
[U7Ø]  157:       }
[9A8]  158:     fprintf(fpvir,"Virus Checker V 1.Ø (c)
          by %s.  ",autor);
[UE7]  159:     fprintf(fpvir,"Unstimmigkeit in Liste!\
          n");
[2Z7]  16Ø:     fprintf(fpvir,"Virusgefahr!! Datei : %s
          ",string);
[KØ9]  161:     fprintf(fpvir,"neue Checksumme: ======>
          >>> Øx%x\n",sum);
[AA2]  162:     fclose(fpvir);
[ØØØ]  163:
[ZM9]  164:     wr_str_pos(15,Ø,"Virusgefahr! Unstimmig
          keit in Liste!",4);
[XM1]  165:     move (16,Ø);
[R39]  166:     printf ("In: %s\a\a\nNeue Checksumme:Øx
          %x",string,sum);
[EB1]  167:     sleep(5);
[EP1]  168:     move(15,Ø);
[851]  169:     cleob();
[9Y1]  17Ø: #ifdef TRACE2
[HBB]  171: fprintf (fptrace,"\npruef: Unstimmigkeit in
           Liste! Virusgefahr!\n");
[T67]  172: fprintf (fptrace,"Datei : %s    \n\n",strin
          g);
[5Z1]  173: #endif TRACE2
[U7Ø]  174:     }
[U7Ø]  175:   }
[NG4]  176: else                  /* Neueintrag */
[P7Ø]  177:   {
[KL6]  178:   fpout=freopen("vircheck.lst","a",fpout);
[3V2]  179:   if (fpout EQ NULL)
[P7Ø]  18Ø:     {
[42C]  181:     printf ("\n\n\a\apruef: 2 VIRCHECK.LST
          läßt sich nicht öffnen!\n\n");
[LF1]  182:     cur_on();
[F61]  183:     exit(2);
[U7Ø]  184:     }
[9Y1]  185: #ifdef TRACE2
[9K9]  186:   fprintf(fptrace,"pruef: Neueintrag! %s %6
          x\n",fname,sum);
[5Z1]  187: #endif TRACE2
[ST5]  188:   fprintf(fpout,"%s %6x\n",fname,sum);
[U7Ø]  189:   }
[ØØØ]  19Ø:
[KC1]  191: return;
[ØP5]  192: }              /* end of function check */
[ØØØ]  193:
[ØG4]  194: /*****************************************/
[ØØØ]  195:
[3M4]  196: void arbeite(dirname,direc)
[ZG6]  197: char  dirname[81];     /* Filedeskriptor */
[9G6]  198: BOOL  direc;   /* dir==TRUE => Directory */
[P7Ø]  199: {
[G91]  2ØØ: int fd;
[3K1]  2Ø1: int hilf;
[5R2]  2Ø2: char datname[81];
[ØF3]  2Ø3: unsigned long hlong;
[ØØØ]  2Ø4:
[5V1]  2Ø5: #ifdef TRACE
[KØB]  2Ø6: fprintf(fptrace,"arbeite: dirname= '%s' dir
          ec= %d\n",dirname,direc);
[1W1]  2Ø7: #endif TRACE
[ØØØ]  2Ø8:
[SK6]  2Ø9: if (NOT direc) /* falls kein Dir => File */
[GZ5]  21Ø:  {              /* Wegen rekursivem Aufruf */
[5X1]  211:  if (ausgabe)
[P7Ø]  212:    {
[YF1]  213:    move(5,8);
[W51]  214:    cleol();
[GA3]  215:    printf("%s",dirname);
[U7Ø]  216:    }
[7L4]  217:  fd=open(dirname,O_RDONLY);
[NW1]  218:  if (fd EQ EOF)
[P7Ø]  219:    {
[UP1]  22Ø:    move(18,Ø);
[ARA]  221:    printf ("\a\aarbeite: '%s' läßt sich nic
          ht öffnen!\n",dirname);
[U7Ø]  222:    }
[ZRØ]  223:  else
[P7Ø]  224:    {
[7P2]  225:    hilf=chksum(fd);
[UK1]  226:    close(fd);
[XØ2]  227:    anz_files++;
[113]  228:    if (NOT neueliste)
[P7Ø]  229:      {
[B93]  23Ø:      pruef(dirname,hilf);
[U7Ø]  231:      }
[NM6]  232:    else /* Nur neue Checkliste erstellen */
[P7Ø]  233:      {
[YD6]  234:      fprintf(fpout,"%s %6x\n",dirname,hilf)
          ;
[U7Ø]  235:      }
[U7Ø]  236:    }
[U7Ø]  237:  }
[ZRØ]  238: else
[P7Ø]  239:   {
[NBØ]  24Ø:   do
[P7Ø]  241:     {
[6P7]  242: fnsplit(ffblock.ff_name,drive,dir,file,suff
          );
[4B5]  243:     if ( (strncmp(suff,".COM",4) EQ Ø) OR
[AV4]  244:          (strncmp(suff,".EXE",4) EQ Ø) OR
```

Listing 1. Das Programm »vircheck.c«, geschrieben in Turbo C, prüft Programmdateien auf Viren

```
[GU4]  245:          (strncmp(suff,".OBJ",4) EQ Ø) OR
[RP4]  246:          (strncmp(suff,".SYS",4) EQ Ø) )
[P7Ø]  247:          {
[794]  248:          strcpy (datname,dirname);
[WM5]  249:          strcat (datname,ffblock.ff_name);
[4Z3]  25Ø:          arbeite(datname,FALSE);
[U7Ø]  251:          }
[ØØØ]  252:
[KW3]  253:     hilf=findnext(&ffblock);
[2X2]  254:          /* get next Datei */
[U7Ø]  255:     }
[XK2]  256:   while (NOT hilf);
[V64]  257:   }                          /* end of else */
[ØØØ]  258:
[KC1]  259: return;
[NK5]  26Ø: }                  /* end function arbeite*/
[ØG4]  261: /*****************************************/
[ØØØ]  262:
[Ø76]  263: void bedir(fdname)       /* Bearbeite Dir */
[7B2]  264: char fdname[];
[P7Ø]  265: {
[3S2]  266: char dirname[81];
[EH2]  267: char pfad [81];
[HN1]  268: int  hilf;
[5V1]  269: #ifdef TRACE
[LA8]  27Ø: fprintf(fptrace,"bedir: parameter= '%s'\n",
            fdname);
[1W1]  271: #endif TRACE
[ØØØ]  272:
[PK8]  273: strcpy(dirname,strtok(fdname," *öäüßÖÄÜ?"))
            ;
[8L3]  274: strcpy(pfad,dirname);
[ØØØ]  275:
[NR6]  276: if (dirname[(strlen(dirname)-1)] NEQ '\\')
[P7Ø]  277:   {
[T83]  278:   strcat(pfad,"\\*.*");
[YD3]  279:   strcat(dirname,"\\");
[U7Ø]  28Ø:   }
[336]  281: else         /* fdname enthält \ am Ende */
[P7Ø]  282:   {
[NV2]  283:   strcat(pfad,"*.*");
[U7Ø]  284:   }
[ØØØ]  285:
[5V1]  286: #ifdef TRACE
[E68]  287: fprintf(fptrace,"bedir: dirname='%s'\n",dir
            name);
[U67]  288: fprintf(fptrace,"bedir: pfad   ='%s'\n",pfa
            d);
[1W1]  289: #endif TRACE
[ØØØ]  29Ø:
[BD8]  291: /* FA_ARCH + FA_HIDDEN + FA_SYSTEM + FA_RDO
            NLY = Øx27 */
[FT5]  292: hilf=findfirst(pfad,&ffblock,Øx27);
[ØØØ]  293:
[5V1]  294: #ifdef TRACE
[LS9]  295: fprintf(fptrace,"bedir: returnwert findfirs
            t= %d\n",hilf);
[1W1]  296: #endif TRACE
[ØØØ]  297:
[S12]  298: if (NOT hilf)
[P7Ø]  299:   {
[BV3]  3ØØ:   arbeite(dirname,TRUE);
[2D6]  3Ø1:  /* Bearbeite alle Dateien in Directory */
[U7Ø]  3Ø2:   }
[KC1]  3Ø3: return;
[U7Ø]  3Ø4: }
[1H3]  3Ø5:                      /* end function bedir */
[ØG4]  3Ø6: /*****************************************/
[ØØØ]  3Ø7:
[ØØØ]  3Ø8:
[5G2]  3Ø9: main(argc,argv)
[NN1]  31Ø: int  argc;
[R42]  311: char *argv[];
[P7Ø]  312: {
[DP5]  313: int    anzahl=Ø,     /* Hilfsvariable  */
[RUØ]  314:        hilf;
[G36]  315: char   dirnam[8Ø];   /* Directory Name */
[3U5]  316: char  *hilfptr;     /* Hilfspointer   */
[MB5]  317: FILE  *fpin;        /* Eingabedatei   */
[ØØØ]  318:
[ZC5]  319: textset();          /* clear screen   */
[NT7]  32Ø: printf("\n  Virus CHECKER  V 1.Ø 1988 by %s
            ",autor);
[ØØØ]  321:
[R61]  322: hilf=Ø;
[8M4]  323: while (isalpha(autor[hilf]))
[P7Ø]  324:   {
[NH4]  325:   anzahl += (int)autor[hilf];
[451]  326:   hilf++;
[U7Ø]  327:   }
[U33]  328: if (anzahl NEQ Øx2cf)
[P7Ø]  329:   {
[YGB]  33Ø:   wr_str_pos(12,15,"Der Programmcode wurde
            unzulässig verändert!",4);
[KS2]  331:   printf("\a\a\a");
[WM1]  332:   move (21,Ø);
[F61]  333:   exit(2);
[U7Ø]  334:   }
[3H1]  335: anzahl=Ø;
[ZP1]  336: cur_off();
[ØØØ]  337:
[VP1]  338: if (argc < 2)
[P7Ø]  339:   {
[3N1]  34Ø:   move (5,Ø);
[4A6]  341:   printf("Fehlerhafte Bedienung! \n\n");
[X7A]  342:   printf("Korrekte Benutzung: virus dateina
            me parameter \n");
[PW9]  343:   printf("oder: virus pfad\\dateiname param
            eter \n\n\a\a");
[5SA]  344:   printf("Die Datei muß eine Liste von Dire
            ctories enthalten,\n");
[3Y6]  345:   printf("die bearbeitet werden sollen.\n")
            ;
[GU9]  346:   printf("Pro Zeile ist ein Directoryeintra
            g zulässig.\n");
[E4A]  347:   printf("Die zulässig maximale Pfadlänge i
            st 68 Bytes.\n\n");
[13A]  348:   printf("Parameter /s => Bildschirmausgabe
             unterdrücken.\n");
[NN9]  349:   printf("Parameter /n => Checkliste wird n
            eu erstellt.\n");
[LF1]  35Ø:   cur_on();
[F61]  351:   exit(2);
[U7Ø]  352:   }
[1R1]  353: if (argc > 2)
[BB5]  354:   {                  /* Parameter /n gesetzt? */
[B7A]  355:   if (argv[2][Ø] EQ '/' AND (argv[2][1] EQ
            'n' OR argv[2][1] EQ 'N'))
[P7Ø]  356:     {
[ZU2]  357:     neueliste=TRUE;
[CE5]  358:     }              /* Parameter /s gesetzt?  */
[48A]  359:   if (argv[2][Ø] EQ '/' AND (argv[2][1] EQ
            's' OR argv[2][1] EQ 'S'))
[P7Ø]  36Ø:     {
[UD2]  361:     ausgabe=FALSE;
[U7Ø]  362:     }
[U7Ø]  363:   }
[2R1]  364: if (argc > 3)
[BB5]  365:   {                  /* Parameter /n gesetzt? */
[L7A]  366:   if (argv[3][Ø] EQ '/' AND (argv[3][1] EQ
            'n' OR argv[3][1] EQ 'N'))
[P7Ø]  367:     {
[ZU2]  368:     neueliste=TRUE;
[CE5]  369:     }              /* Parameter /s gesetzt?  */
[C8A]  37Ø:   if (argv[3][Ø] EQ '/' AND (argv[3][1] EQ
            's' OR argv[3][1] EQ 'S'))
[P7Ø]  371:     {
[UD2]  372:     ausgabe=FALSE;
[U7Ø]  373:     }
[U7Ø]  374:   }
[892]  375: if (neueliste)
[P7Ø]  376:   {
[GX8]  377:   printf("\n\n Es wird eine neue Checkliste
             erstellt!");
[U7Ø]  378:   }
[ØØØ]  379:
[5X1]  38Ø: if (ausgabe)
[P7Ø]  381:   {
[E94]  382:   wr_str_pos(5,Ø,"File: ",Ø);
[E52]  383:   box(7,9,25,52);
[ZT5]  384:   wr_str_pos(8,26,"Check_Summe: ",Ø);
[U7Ø]  385:   }
[ØØØ]  386:
[5V1]  387: #ifdef TRACE
[FN5]  388: fptrace=fopen("virtrace.out","w");
[V63]  389: if (fptrace EQ NULL)
[P7Ø]  39Ø:   {
[5CC]  391:   printf("\n\n\a\aVIRTRACE.OUT läßt sich ni
            cht öffnen/erzeugen!\n\n");
[LF1]  392:   cur_on();
[F61]  393:   exit(2);
[U7Ø]  394:   }
[1W1]  395: #endif TRACE
[ØØØ]  396:
[9M6]  397: /* Neue Liste erstellen: alte L.sichern*/
[892]  398: if (neueliste)
[P7Ø]  399:   {
[ST3]  4ØØ:   unlink("vircheck.bck");
[C46]  4Ø1:   rename("vircheck.lst","vircheck.bck");
[U7Ø]  4Ø2:   }
[ØØØ]  4Ø3:
[X25]  4Ø4: fpout=fopen("vircheck.lst","a");
[3V2]  4Ø5: if (fpout EQ NULL)
[P7Ø]  4Ø6:   {
[Y7C]  4Ø7:   printf ("\n\n\a\aVIRCHECK.LST läßt sich n
            icht öffnen/erzeugen!\n\n");
[LF1]  4Ø8:   cur_on();
[F61]  4Ø9:   exit(2);
[U7Ø]  41Ø:   }
[ØØØ]  411:
[VV3]  412: fpin=fopen(argv[1],"r");
[PP2]  413: if (fpin EQ NULL)
[P7Ø]  414:   {
[SZB]  415:   printf("\n\n\a\amain: Eingabedatei %s läß
            t sich nicht öffnen!\n\n", \
```

```
[6K1]  416:         argv[1]);
[LF1]  417:    cur_on();
[F61]  418:    exit(2);
[U7Ø]  419:    }
[ØØØ]  42Ø:
[TD6]  421: /* if Virus.lst exist => viruslst.bak */
[YX3]  422: unlink("viruslst.bck");
[ØU5]  423: rename("virus.lst","viruslst.bck");
[ØØØ]  424:
[UT5]  425:       /* Abarbeitung der Directory-Liste */
[UM1]  426: move (13,Ø);
[NBØ]  427: do
[P7Ø]  428:    {
[VZ4]  429:    hilfptr=fgets(dirnam,7Ø,fpin);
[393]  43Ø:    if (hilfptr EQ NULL)
[P7Ø]  431:      {
[PØ1]  432:      break;
[U7Ø]  433:      }
[R61]  434:    hilf=Ø;
[6G5]  435:    while (NOT iscntrl(dirnam[hilf]))
[P7Ø]  436:      {
[451]  437:      hilf++;
[U7Ø]  438:      }
[MR6]  439:    dirnam[hilf]='\Ø'; /* Stringendesetzen */
[ØØØ]  44Ø:
[5V1]  441: #ifdef TRACE
[ØK7]  442: fprintf(fptrace,"main: dirnam= '%s'\n",dirn
       am);
[1W1]  443: #endif TRACE
[UC6]  444:    bedir(dirnam); /* Directory bearbeiten */
[PE1]  445:    anzahl++;
[YM2]  446:    if (NOT ausgabe)
[P7Ø]  447:      {
[PØ6]  448:      printf("."); /* Punktausgabe pro Dir */
[U7Ø]  449:      }
[U7Ø]  45Ø:    }
[A12]  451: while (TRUE);
[ØØØ]  452:
[142]  453: fclose(fpin);
[7B2]  454: fclose(fpout);
[ØØØ]  455:
[HN5]  456: textset();                  /* clear screen */
[YP7]  457: printf("\n   Virus CHECKER V 1.Ø 1988 by %s
       ",autor);
[7N1]  458: move (6,Ø);
[GPA]  459: printf("Es wurden insgesamt %u Dateien bear
       beitet\n",anz_files);
[FN1]  46Ø: move (8,Ø);
[XWA]  461: printf("Es wurden insgesamt %d Directories
       bearbeitet\n",anzahl);
[HK1]  462: if (virus)
[P7Ø]  463:    {
[HXB]  464:    wr_str_pos(1Ø,Ø,"Virusgefahr! Unstimmigke
       it in Liste! Virusgefahr!",4);
[TM1]  465:    move (12,Ø);
[4XA]  466:    printf("\a\aEs ist eine Checksummendiffer
       enz ermittelt worden!");
[KDA]  467:    printf("\n\nBitte sehen Sie in die Datei
       VIRUS.LST!\a\a\n\n");
[U7Ø]  468:    }
[ØØØ]  469:
[LF1]  47Ø: cur_on();
```

Listing 1. Das Programm »vircheck.c«, geschrieben in Turbo C, prüft Programmdateien auf Viren

```
[W6Ø]    1: /*
[VD6]    2:    Positions-Routinen für den Textmodus
[LN3]    3:    mit Attributauswahl
[ØØØ]    4:
[TH2]    5:    by B.Reklies 1987
[ØØØ]    6:
[17Ø]    7: */
[ØØØ]    8:
[9L2]    9: #include <dos.h>
[ØØØ]   1Ø:
[K93]   11: #define VIDEO       Øx1Ø
[ØØØ]   12:
[263]   13: #define STR_IN      15
[FN3]   14:                     /* intensive Schrift */
[V53]   15: #define INVERS     112
[G33]   16:                       /* Inverse Schrift */
```

```
[FB3]   17: #define STR_BLK    135
[ZY2]   18:                       /* Blinken normal  */
[6M3]   19: #define STR_BLK_IN 143
[5D3]   2Ø:                       /* Blinken intensiv */
[DA3]   21: #define INV_BLK    24Ø
[Z63]   22:                       /*  Invers blinken  */
[ØØØ]   23:
[ØØØ]   24:
[LU2]   25:                       /*   move Cursor   */
[ØØØ]   26:
[HB2]   27: void move(z,s)
[TW5]   28: unsigned z,s;           /* Zeile, Spalte */
[ØØØ]   29:
[P7Ø]   3Ø: {
[FP2]   31:   union REGS regs;
[ØØØ]   32:
[WE1]   33:   if (z > 24)
[MØ1]   34:     z = 24;
[ØØØ]   35:
[7F1]   36:   if (s > 79)
[ZØ1]   37:     s = 79;
[ØØØ]   38:
[812]   39:   regs.h.ah = 2;
[N32]   4Ø:   regs.h.dl = s;
[V32]   41:   regs.h.dh = z;
[512]   42:   regs.h.bh = Ø;
[324]   43:   int86(VIDEO,&regs,&regs);
[KC1]   44:   return;
[U7Ø]   45: }
[ØØØ]   46:
[UR4]   47:              /* clear screen / Textmodus */
[ØØØ]   48:
[XB2]   49: void textset()
[P7Ø]   5Ø: {
[FP2]   51:   union REGS regs;
[ØØØ]   52:
[412]   53:   regs.h.ah = Ø;
[K12]   54:   regs.h.al = 3;
[324]   55:   int86(VIDEO,&regs,&regs);
[KC1]   56:   return;
[U7Ø]   57: }
[ØP4]   58:              /* write char / move Cursor */
[ØØØ]   59:
[T92]   6Ø: void wr_ch(ch)
[S81]   61: char ch;
[P7Ø]   62: {
[FP2]   63:   union REGS regs;
[ØØØ]   64:
[X72]   65:   regs.h.ah = 14;
[5B2]   66:   regs.h.al = ch;
[324]   67:   int86(VIDEO,&regs,&regs);
[KC1]   68:   return;
[U7Ø]   69: }
[5V3]   7Ø:                  /* clear to end of line */
[ØØØ]   71:
[AY1]   72: void cleol()
[P7Ø]   73: {
[FP2]   74:   union REGS regs;
[VA2]   75:   int i,z,s,end;
[ØØØ]   76:
[A12]   77:   regs.h.ah = 3;
[512]   78:   regs.h.bh = Ø;
[324]   79:   int86(VIDEO,&regs,&regs);
[V32]   8Ø:   z = regs.h.dh;
[ZU5]   81:   s = regs.h.dl;       /* Position merken */
[ØØØ]   82:
[YM1]   83:   end = 8Ø - s;
[EY2]   84:   for (i=Ø;i<end;i++)
[CL1]   85:     wr_ch(32);
[ØØØ]   86:
[T26]   87:   move (z,s);         /* Rückkehrposition */
[KC1]   88:   return;
[U7Ø]   89: }
[M64]   9Ø:                /* clear to end of bottom */
[ØØØ]   91:
[CX1]   92: void cleob()
[P7Ø]   93: {
[FP2]   94:   union REGS regs;
[VA2]   95:   int i,z,s,end;
[ØØØ]   96:
[A12]   97:   regs.h.ah = 3;
[512]   98:   regs.h.bh = Ø;
[324]   99:   int86(VIDEO,&regs,&regs);
[V32]  1ØØ:   z = regs.h.dh;
[ZU5]  1Ø1:   s = regs.h.dl;       /* Position merken */
[ØØØ]  1Ø2:
[KC3]  1Ø3:   end = ((24-z)*8Ø) + 8Ø-s-1;
[TP4]  1Ø4:         /* -1, damit er nicht scrollt */
[EY2]  1Ø5:   for (i=Ø;i<end;i++)
[CL1]  1Ø6:     wr_ch(32);
[ØØØ]  1Ø7:
[T26]  1Ø8:   move (z,s);         /* Rückkehrposition */
```

Listing 2. Die Header-Datei »grafik.h« bietet schnelle Textausgaberoutinen, die den Virusprüfer von der Version des Turbo-C-Compilers unabhängig machen

```
[KC1]  109:   return;
[U70]  110: }
[000]  111:
[392]  112:                                /* cursor off */
[000]  113:
[4C2]  114: void cur_off()
[P70]  115: {
[FP2]  116:   union REGS regs;
[000]  117:
[612]  118:   regs.h.ah = 1;
[282]  119:   regs.h.ch = 15;
[A82]  120:   regs.h.cl = 15;
[324]  121:   int86(VIDEO,&regs,&regs);
[KC1]  122:   return;
[U70]  123: }
[000]  124:
[272]  125:                                /* cursor on */
[000]  126:
[D82]  127: void cur_on()
[P70]  128: {
[FP2]  129:   union REGS regs;
[000]  130:
[612]  131:   regs.h.ah = 1;
[T72]  132:   regs.h.ch = 12;
[A82]  133:   regs.h.cl = 15;
[324]  134:   int86(VIDEO,&regs,&regs);
[KC1]  135:   return;
[U70]  136: }
[000]  137:
[000]  138:
[1R3]  139:                    /* write char blink on */
[000]  140:
[VF3]  141: void wr_chbl(ch,att)
[805]  142: char     ch;                  /* Zeichen  */
[5S5]  143: unsigned att;                 /* Attribut */
[P70]  144: {
[1S6]  145:   unsigned zalt,salt;  /* Postion merken */
[FP2]  146:   union REGS regs;
[000]  147:
[A12]  148:   regs.h.ah = 3;
[512]  149:   regs.h.bh = 0;
[324]  150:   int86(VIDEO,&regs,&regs);
[5M2]  151:   zalt = regs.h.dh;
[VL2]  152:   salt = regs.h.dl;
[000]  153:
[P12]  154:   regs.h.ah = 9;
[512]  155:   regs.h.bh = 0;
[S32]  156:   regs.x.cx = 1;
[5B2]  157:   regs.h.al = ch;
[5H2]  158:   regs.h.bl = att;
[324]  159:   int86(VIDEO,&regs,&regs);
[663]  160:   move (zalt,salt+1);
[DM4]  161:                /* gehe zur neuen Position */
[KC1]  162:   return;
[U70]  163: }
[000]  164:
[TV4]  165:              /* write string on position */
[000]  166:
[FB5]  167: void wr_str_pos(z,s,str,attr)
[TA5]  168: unsigned z;                 /*   zeile      */
[4C5]  169: unsigned s;                 /*   spalte     */
[ND5]  170: char str[];                 /*   string     */
[196]  171: unsigned attr;              /* Attributwahl */
[000]  172:
[P70]  173: {
[1S6]  174:   unsigned zalt,salt;  /* Postion merken */
[CY5]  175:   unsigned att;         /* Attributwert   */
[WP5]  176:   unsigned i = 0;       /* Laufvariable   */
[FP2]  177:   union REGS regs;
[000]  178:
[A12]  179:   regs.h.ah = 3;
[512]  180:   regs.h.bh = 0;
[324]  181:   int86(VIDEO,&regs,&regs);
[000]  182:
[5M2]  183:   zalt = regs.h.dh;
[VL2]  184:   salt = regs.h.dl;
[AH6]  185:   move (z,s); /* gehe zur neuen Position */
[000]  186:
[V42]  187:   switch (attr)
[P70]  188:    {
[RK3]  189:    case  1 : att = STR_IN;
[P01]  190:              break;
[CK3]  191:    case  2 : att = INVERS;
[P01]  192:              break;
[6P3]  193:    case  3 : att = STR_BLK;
[P01]  194:              break;
[PA4]  195:    case  4 : att = STR_BLK_IN;
[P01]  196:              break;
[SN3]  197:    case  5 : att = INV_BLK;
[P01]  198:              break;
[U70]  199:    }
[000]  200:
[313]  201:   if (attr>0 && attr <6)
[P70]  202:     {
[Z83]  203:     while (str[i] != '\0')
[P70]  204:       {
[LL3]  205:       wr_chbl(str[i],att);
[UF0]  206:       i++;
[U70]  207:       }
[U70]  208:     }
[ZR0]  209:   else
[P70]  210:     {
[Z83]  211:     while (str[i] != '\0')
[P70]  212:       {
[4K2]  213:       wr_ch(str[i]);
[UF0]  214:       i++;
[U70]  215:       }
[U70]  216:     }
[HV2]  217:   move (zalt,salt);
[4M4]  218:              /* gehe zur alten Position */
[KC1]  219:   return;
[U70]  220: }
[000]  221:
[X65]  222:           /* write string with attribut */
[000]  223:
[7W3]  224: void wr_str(str,attr)
[ND5]  225: char str[];                 /*   string     */
[G66]  226: unsigned attr               /* Attributwahl */
[000]  227:
[P70]  228: {
[4X5]  229:   unsigned att;             /* Attributwert */
[AN5]  230:   unsigned i = 0;           /* Laufvariable */
[FP2]  231:   union REGS regs;
[000]  232:
[V42]  233:   switch (attr)
[P70]  234:    {
[RK3]  235:    case  1 : att = STR_IN;
[P01]  236:              break;
[CK3]  237:    case  2 : att = INVERS;
[P01]  238:              break;
[6P3]  239:    case  3 : att = STR_BLK;
[P01]  240:              break;
[PA4]  241:    case  4 : att = STR_BLK_IN;
[P01]  242:              break;
[SN3]  243:    case  5 : att = INV_BLK;
[P01]  244:              break;
[U70]  245:    }
[000]  246:
[313]  247:   if (attr>0 && attr <6)
[P70]  248:     {
[Z83]  249:     while (str[i] != '\0')
[P70]  250:       {
[LL3]  251:       wr_chbl(str[i],att);
[UF0]  252:       i++;
[U70]  253:       }
[U70]  254:     }
[ZR0]  255:   else
[P70]  256:     {
[Z83]  257:     while (str[i] != '\0')
[P70]  258:       {
[4K2]  259:       wr_ch(str[i]);
[UF0]  260:       i++;
[U70]  261:       }
[U70]  262:     }
[KC1]  263:   return;
[U70]  264: }
[000]  265:
[000]  266:
[952]  267:                                /* paint box */
[000]  268:
[SC3]  269: void box(za,ze,sa,se)
[UM5]  270: unsigned za,ze;                  /* zeile  */
[RN5]  271: unsigned sa,se;                  /* spalte */
[000]  272:
[P70]  273: {
[XK5]  274:   unsigned i;               /* Laufvariable */
[000]  275:
[UZ1]  276:   move(za,sa);
[CT1]  277:   wr_ch(201);
[653]  278:   for (i=sa+1;i<se;i++)
[MT1]  279:     wr_ch(205);
[WT1]  280:   wr_ch(187);
[000]  281:
[963]  282:   for (i=za+1;i<ze;i++)
[P70]  283:     {
[VT1]  284:     move(i,sa);
[UT1]  285:     wr_ch(186);
[3U1]  286:     move(i,se);
[UT1]  287:     wr_ch(186);
[U70]  288:     }
[000]  289:
[A02]  290:   move(ze,sa);
[AT1]  291:   wr_ch(200);
[653]  292:   for (i=sa+1;i<se;i++)
[MT1]  293:     wr_ch(205);
[YT1]  294:   wr_ch(188);
[000]  295:
[KC1]  296:   return;
[U70]  297: }
```

Listing 2. Die Header-Datei »grafik.h« bietet schnelle Textausgaberoutinen, die den Virusprüfer von der Version des Turbo-C-Compilers unabhängig machen

Praxiserprobte Basic-Musterprogramme

inklusive Diskette

Erst prüfen, dann kaufen:
Sehen Sie sich dieses Werk in Ruhe an: 10 Tage lang dürfen Sie Ihr Ansichtsexemplar unverbindlich zu Hause prüfen.

Basic-Tools, Utilities und Anwenderprogramme für Wirtschaft, Technik, Mathematik

Dieses Nachschlagewerk bietet Ihnen:
auf ca. 1100 Seiten sofort einsetzbare Programme und Utilities für PC-/MS-DOS-Computer. Hinweise auf gerätespezifische Besonderheiten ermöglichen Ihnen auch die problemlose Anpassung an andere Rechner. Sie erhalten u. a.

■ **anspruchsvolle Anwenderprogramme wie**

– das leistungsstarke Textverarbeitungsprogramm „SOTEXT 4.0“, komplett menügesteuert und mit deutschen Befehlen für den problemlosen Soforteinsatz

– SOTEXT 4.0 zusätzlich als sofort lauffähige Version und in Sourcecode auf Diskette!

– Programme für Wirtschaft und Verwaltung: Finanzbuchhaltung „SOBUCH“, Netzplantechnik, Terminkalender, Berechnung der Einkommensteuer, Rabattermittlung, Tilgungsplan eines Darlehens, Restwertberechnung ...;

– mathematische Programme: Analyse von Funktionsgleichungen (z. B. Kurvendiskussion; lineare Algebra: z. B. Matrizenoperationen, Simplexalgorithmus);

– statistische Programme (u. a. Berechnung von Mittelwert, Streuung, Standardabweichung);

– technische Programme: Messen/Steuern/Regeln, Wertetabellen von Digitalschaltungen, Berechnen von Querschnittswerten ...;

Programme für die Ausbildung (z. B. Vokabeltraining);

■ **Tools und vielseitige Routinen für**

– Grafik: Grafik-Generator „SOGRAF“, Zeichengenerator;

– Dateiverwaltung und Dateiorganisation: Dateigenerator, Komprimieren und Expandieren von Textdateien, Konvertieren von Dateien, Disketten-Editor „SOZAP“, indexsequentielle Dateiorganisation und Reorganisation, Hash-Dateiorganisation; Such- und Sortierverfahren;

– Drucker (z. B. Ausgabe von Random-Dateien mit Zeilenvorschub, Druckereinstellung);

– Bildschirm (z. B. Bildschirm- und Menütechnik);

– Multitasking (z. B. Programm zur Parallelverarbeitung);

Jedes Programm ist ausführlich dokumentiert mit Problembeschreibung, Vorgehensweise, Struktogramm, Listing und Probelauf.

Fordern Sie noch heute an:

Aktuelle Musterprogramme in Basic

stabiler Ringbuchordner, DIN A4, über 1100 Seiten, inkl. Diskette „SOTEXT 4.0“, Bestell-Nr. 1300/71300, Preis: DM 98,–.
Alle 2 – 3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 130 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).

INTEREST-VERLAG
Industriestraße 21
D-8901 Kissing
Tel. 0 82 33 / 21 01-0

Benutzeroberfläche und Präprozessor für SBasic

Der SBasic-Compiler aus der DOS-Extra-Ausgabe 6 übersetzt bekanntlich nur Programme, die mit einem GW-Basic-Interpreter im Token-Format gespeichert sind. Aus diesem Grund mußte man sich bei der Programmierung bislang an die strengen GW-Basic-Normen halten. Doch damit ist es nun vorbei!

Herkömmlichen GW-Basic-Quelltexten mangelt es stets an Übersichtlichkeit. Jeder Zeile muß eine Nummer voranstehen, damit der Interpreter sie akzeptiert. Muß man Zeilen einfügen, löschen oder überschreiben, kann es unter Umständen zu Problemen kommen, wenn die Abstände der Zeilennummern zu gering sind. Ein neu numeriertes Programm verwirrt dann nur noch, weil man sich die alten Zeilennummern bereits eingeprägt hat.

Professionelle Basic-Compiler sind deshalb schon lange dazu übergegangen, auch Quelltexte zu übersetzen, deren Verzweigungen sich auf Marken (Labels) beziehen. Aus diesem Grund wurde auch für den SBasic-Compiler ein Präprozessor entwickelt, der ebenfalls die Verwendung von Marken erlaubt. Nebenbei erhalten Sie auch eine Benutzeroberfläche, die die Programmentwicklung mit SBasic ganz erheblich vereinfacht.

Bei dem Programm »sbshell.bas« (Listing 1) handelt es sich um den Kern der SBasic-Benutzeroberfläche, aus dem Sie alle benötigten Funktionen auswählen (Bild 1). Der Präprozessor »preproz.bas« (Listing 2) wandelt Quelltexte, deren Verzweigungen sich auf Sprungmarken beziehen (wie in Listing 3) in das für GW-Basic verträgliche Format mit Zeilennummern um.

Die Marken dürfen bis zu 32 Zeichen lang sein und Buchstaben, Zahlen sowie den Unterstrich (»_«) beinhalten. Dazu ein Beispiel:

```
ende: cls:end
```

Diese Programmzeile vereinbart die Sprungmarke »ende« an dieser Position in einem Programm. Die Anweisung

```
goto ende
```

Bild 1. Das neue Gesicht von SBasic ist zwar klein – aber fein

bewirkt dadurch eine unbedingte Verzweigung zur Marke »ende«, an der dann der Bildschirm gelöscht und das Programm beendet wird. Die Marke muß übrigens nicht unbedingt am Zeilenanfang sein, ihr dürfen aber nur Leerzeichen voranstehen. Ebenso muß ihr sofort ein Doppelpunkt folgen, und wenn sich in der gleichen Zeile noch eine Basic-Anweisung befindet, steht vor dieser ein Leerzeichen.

Der Präprozessor bearbeitet den Basic-Quelltext in zwei Durchgängen. Zuerst ordnet er jeder Zeile eine Nummer zu und speichert diejenigen Zeilen, die mit einer Marke beginnen, zusammen mit der Marke auf Datenträger ab. Die Numerierung erfolgt mit der Schrittweite 1, damit sich Fehlermeldungen des Programms oder des Compilers auch auf im Editor nachvollziehbare Zeilen beziehen. Aus diesem Grund wandelt der Präprozessor auch Zeilen, die nur aus einer Marke bestehen, in Kommentarzeilen um.

Steht eine Marke nicht allein, folgen ihr also eine oder mehrere Basic-Anweisungen, so fügt der Präprozessor im ersten Durchgang den Namen der Marke als Bemerkung an die Programmzeile an und erzeugt eine Datei mit dem Namen des Quelltextes und der Endung »$$$«.

Im zweiten Durchgang untersucht der Präprozessor jede Zeile nach den Schlüsselwörtern »goto«, »gosub«, »restore« und »then«. Den Befehlen »run« und »list« dürfen zwar üblicherweise ebenfalls Zeilennummern folgen, doch arbeitet SBasic ohnehin nicht damit. Der Then-Befehl bezieht sich unter Umständen ebenfalls auf eine Marke, weshalb Sie keine Marken verwenden dürfen, die wie ein alleinstehender Befehl lauten, also zum Beispiel »print« oder »wend«.

Die vom Präprozessor erzeugte Datei hat nach dem zweiten Durchgang die Endung »bas«. Damit der mit Sprungmarken durchsetzte Ur-Quelltext nicht von dieser überschrieben wird, muß er also eine andere Endung besitzen. Der Präprozessor und die Benutzeroberfläche gehen deshalb davon aus, daß er mit »asc« (für ASCII-Format) endet.

Zuletzt generiert der Präprozessor ein kleines Basic-Programm, welches das erzeugte Programm in den GW-Basic-Interpreter lädt, im Token-Format speichert und den Interpreter wieder verläßt.

Die Benutzeroberfläche von SBasic stellt die Funktionen »Dateiname«, »Editor«, »Compiler«, »Start«, »GW-Basic«, »Betriebssystem« und »Verlassen« bereit. Weil SBasic kaum Fehlerabfangmechanismen besitzt, überläßt es die meisten Überprüfungen verschiedenen Stapeldateien, die Sie mit dem Programm »makebat.asc« (Listing 3) erzeugen.

Gehen Sie zum Erzeugen der einzelnen Programme wie folgt vor:

Tippen Sie »sbshell.bas« (Listing 1) und »preproz.bas« (Listing 2) ab, und wandeln Sie beide Programme mit

```
gwbasic
[F3]sbshell
[F4]sbshell
[F3]preproz
[F4]preproz
system
```

über den GW-Basic-Interpreter in das Token-Format um. Anschließend compilieren Sie die Dateien mit SBasic. Geben Sie darauf »makebat.asc« (Listing 3) ohne Zeilennummern in einen Editor ein, und wandeln Sie den Quelltext mit dem compilierten Präprozessor »preproz.exe« in ein GW-Basic-kompatibles Format um.

Der Aufruf

```
preproz makebat
```

erzeugt übrigens die beiden temporären Dateien »$$$.bas« und »makebat.$$$«, die Sie von Hand löschen müssen. Später übernimmt die SBasic-Benutzeroberfläche das Löschen von temporären Dateien, so daß dieser Arbeitsgang nur einmal notwendig ist.

Es steht Ihnen dann frei, ob Sie das erzeugte Programm »makebat.bas« gleich im Basic-Interpreter ablaufen lassen oder es nach obigen Muster in das Token-Format wandeln und compilieren.
Nachdem Sie »makebat« gestartet haben, erzeugt das Programm fünf Stapeldateien (»start.bat«, »edit.bat«, »comp.bat«, »basic.bat« und »ende.bat«), die von dem Benutzeroberflächenprogramm »sbshell« benötigt werden. Als letztes tippen Sie die kurze Stapeldatei »sb.bat« (Listing 4) ab, und Ihr neues SBasic-System ist komplett. Gestartet wird das System mit

```
sb <Dateiname>
```

Der angegebene Name ist der eines Basic-Programms ohne Erweiterung, Sie sehen ihn nach dem Start der Benutzeroberfläche zur Kontrolle in der linken oberen Ecke des Bildschirms.
Möchten Sie ein anderes Programm bearbeiten, so aktivieren Sie den Menüpunkt »Dateiname«, indem Sie den Menübalken mit den Cursor-Tasten auf diesem Menüpunkt positionieren und die Enter-Taste drücken oder den Anfangsbuchstaben des Menüpunktes, in diesem Fall also [D] eingeben. Rechts öffnet sich darauf ein Fenster, in dem Sie den aktuellen Dateinamen sehen und editieren können. Die Taste [Esc] verläßt das Fenster ohne Änderungen, und mit [Enter] bestätigen Sie den neuen Dateinamen. Darauf passiert im einzelnen folgendes: »sbshell« erzeugt eine Stapeldatei mit Namen »sb1.bat« und dem Inhalt

```
sb <Neuer Name>
```

und beendet das Programm. Die Stapeldatei »sb.bat«, die Sie zuvor aufriefen, endet mit

```
sb1 %1
```

führt also die Arbeit mit der eben erst erzeugten Stapeldatei fort. Diese ruft wiederum die erste Stapeldatei mit dem neuen Dateinamen auf, die »sbshell« startet, der dieser neue Name jetzt auch bekannt ist. Dieses Verfahren ist die einzige Chance, um dem aktuellen Übergabeparameter »%1« den neuen Namen zuzuweisen, ohne dabei »sb.bat« zu ändern. Die Stapeldatei würde nämlich ihre Arbeit einstellen, wenn sie zwischenzeitlich modifiziert worden wäre. Auf ähnlichem Weg arbeiten auch fast alle anderen Funktionen. Um einen Quelltext zu bearbeiten, wählen Sie den Menüpunkt »Editor« an. »sb« kopiert jetzt die Stapeldatei »edit.bat« auf die Datei »sb1.bat« und endet. Wie zuvor ruft die aktuelle Stapeldatei wieder »sb1.bat« auf, wo der Editor

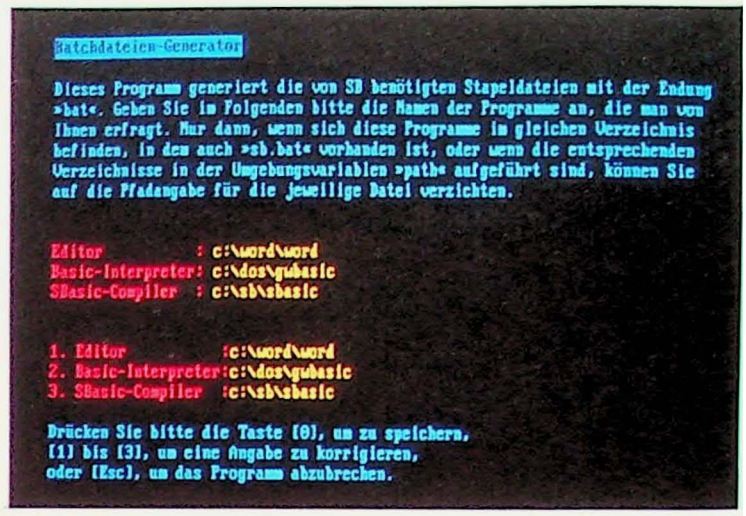


Bild 2. Mit »makebat« richten Sie Ihre SBasic-Entwicklungsumgebung ein

mit dem Dateinamen »%1.asc« also dem übergebenen Dateinamen und der Endung »asc« aufgerufen wird. Mit Beenden des Editors ist wieder »sb.bat« an der Reihe.

Der Menüpunkt »Compiler« kopiert »comp.bat« auf »sb1.bat«. Diese Stapeldatei muß zuerst überprüfen, ob der benötigte Quelltext auch wirklich vorhanden ist und ruft danach den Präprozessor auf. Hat dieser seine Arbeit beendet, so muß das Programm »$$$.bas« generiert worden sein, sonst ist ein Fehler unterlaufen. Darauf startet die Spapeldatei den GW-Basic-Interpreter mit dem Programm »$$$.bas«, das wiederum das umgewandelte Programm lädt und im Token-Format speichert. »$$$.bas« definiert sich dazu die Funktionstasten selbst und schreibt sie anschließend in den Tastaturpuffer.

Sobald »$$$.bas« seine Aufgabe beendet hat, kehrt der Interpreter wie von Geisterhand bewegt in das Betriebssystem zurück. Das alles sehen Sie natürlich nicht, weil die aktuelle Schriftfarbe auf »Schwarz auf Schwarz« eingestellt, also unsichtbar ist. Interessiert Sie aber der beschriebene Vorgang, dann ändern Sie einfach die Einstellung ab der Markenzeile »prinit« in »preproz.bas«. Ebenso können Sie die Arbeit der Stapeldateien verfolgen, wenn Sie aus »sb.bat« den ersten Eintrag »echo off« entfernen. Hat Ihr Basic-Programm endlich das Token-Format ange-

Toolboxen für Programmierer gesucht

Haben Sie eine Toolbox in C, Pascal, Basic, dBase, Clipper oder Assembler geschrieben? Beispielsweise schnelle, universelle Grafikroutinen, eine Textverarbeitungs- oder eine Datenbank-Toolbox? Dann sollten Sie sich schnellstmöglich mit uns in Verbindung setzen. Schicken Sie uns unverbindlich eine Diskette mit Ihrem Programm und einen Ausdruck der Dokumentation.
Falls Sie sich vorweg informieren möchten: Zwischen 10.00 und 12.00 Uhr und zwischen 15.00 und 17.00 Uhr stehen wir Ihnen an allen Werktagen unter der Rufnummer (089) 4 61 00 10 zur Verfügung.

Schicken Sie Ihre Unterlagen an folgende Adresse:
Redaktionsbüro Everts & Hagedorn
»DOS Software«
Hans-Stießberger-Str. 3
8013 Haar/München

nommen, übernimmt SBasic und compiliert das Programm. Anschließend befinden Sie sich wieder in der Benutzeroberfläche, wo Sie das Programm editieren können, falls es Fehler hat, oder über »Start« ausführen. Verarbeitet Ihr Programm Parameter, die beim Aufruf übergeben werden, so müssen Sie den Menüpunkt »Betriebssystem« verwenden. Dort können Sie auch einen MS-DOS-Befehl eingeben, um beispielsweise den Inhalt des Datenträgers zu listen. Der Betriebssystem-Menüpunkt ist übrigens der einzige Menüpunkt, der keine Stapeldatei zur Ausführung erfordert.

Sind Ihnen bei der Programmierung logische Fehler unterlaufen, können Sie diese am besten mit dem GW-Basic-Interpreter ermitteln. Wählen Sie dazu den Menüpunkt »GW-Basic« an. Dieser lädt zunächst wieder mit Hilfe des Tastaturpuffers Ihr Basic-Programm, das jetzt Zeilennummern hat. Beachten Sie, daß die Systemvariablen »command$(n)« von SBasic im Interpreter unbelegt sind. Verwenden Sie sie im Programm, so müssen Sie den Variablen im Direktmodus sinnvolle Werte zuweisen und das Programm anschließend mit dem Goto-Befehl starten.

Vor allem dann, wenn ein compiliertes Programm vorzeitig abgebrochen wurde, bekommen Sie im Interpreter eine aussagekräftige Fehlermeldung, die sich auf eine Zeilennummer bezieht, die in Ihrem Quelltext identisch ist. Nach Eingabe des System-Befehls befinden Sie sich nach erfolgreicher Fehlersuche wieder in der SBasic-Benutzeroberfläche.

Das Hin und Her der Stapeldateien hat übrigens durchaus seinen Sinn. Würden sich die Programme nämlich gegenseitig aufrufen, so könnten sie sich im ungünstigsten Fall im Arbeitsspeicher des PC überlagern. Besäßen Sie noch dazu »nur« 512 KByte davon, so bekämen Sie sehr bald Speicherplatzprobleme. Durch das Stapeldateiverfahren sind aber nur maximal 300 KByte Speicherplatz gebunden. Ist Ihnen selbst das zuviel, dann müssen Sie die maximale Anzahl der Sprungmarken (Labels) in der Zeile 6 und die Sicherheitsabfrage in Zeile 36 des Programms »preproz.bas« auf einen geringeren Wert ändern.

(Dieter Bäumler/ma)

Name	sbshell.bas, preproz.bas, makebat.asc
Sprache	SBasic
Grafikkarte	jede
Besonderheiten	keine

```
[RH4] 1 'Programmname: sbshell.bas
[8M6] 2 '(c) 1989 Redaktionsbüro Everts&Hagedorn
[M63] 3 'DOS International
[934] 4 'Autor: Dieter Bäumler
[6BØ] 5 '
[7M4] 1Ø DIM A$(7):GOSUB 4ØØØ:GOSUB 3ØØØ
[RS1] 2Ø GOSUB 5ØØØ
[WS1] 3Ø GOSUB 6ØØØ
[7Ø5] 4Ø A$=INKEY$:IF A$="" THEN GOTO 4Ø
[GB4] 5Ø IF A$=CHR$(13) THEN GOTO 1ØØ
[NØ4] 6Ø IF ASC(A$)=Ø THEN GOTO 95Ø
[DU3] 7Ø A=INT(INSTR(T$,A$)/2+.5)
[DØ4] 8Ø IF A=Ø THEN BEEP:GOTO 4Ø
[EM3] 9Ø IA=IN:IN=A:GOSUB 6ØØØ
[H55] 1ØØ ON IN-1 GOTO 4ØØ,5ØØ,6ØØ,7ØØ,8ØØ,9ØØ
[BM3] 2ØØ COLOR 11,1:LOCATE 2,48
[MØ5] 21Ø PRINT KO$;STRING$(1Ø,WA$);RO$
[FG5] 22Ø LOCATE 3,59:PRINT SE$:LOCATE 4,48
[XØ5] 23Ø PRINT KL$;STRING$(1Ø,WA$);RU$
[SG4] 24Ø U$=DAT$:L=8:Y=3:X=5Ø:Z$=ZD$
[8T3] 25Ø GOSUB 2ØØØ:LOCATE ,,Ø,Ø
[U95] 26Ø IF ESC=1 THEN GOSUB 5ØØ5:GOTO 3Ø
[A85] 27Ø LT=LEN(U$):IF LT=Ø THEN GOTO 32Ø
[MR2] 28Ø PR$=RIGHT$(U$,1)
[GW3] 29Ø IF PR$<>" " THEN GOTO 32Ø
[KY2] 3ØØ U$=LEFT$(U$,LT-1)
[NP1] 31Ø GOTO 27Ø
[XA4] 32Ø DAT$=U$:LD=INSTR(DAT$," ")
[AB4] 33Ø IF LD>Ø THEN BEEP:GOTO 24Ø
[CP4] 34Ø IF DAT$="" THEN DAT$="NONAME"
[WY4] 35Ø SHELL"echo sb "+DAT$+">sb1.bat"
[EU2] 36Ø LOCATE 13,1:END
[RT5] 4ØØ SHELL"copy edit.bat sb1.bat>nul"
[1U2] 41Ø LOCATE 13,1:END
[MT5] 5ØØ SHELL "copy comp.bat sb1.bat>nul"
[3U2] 51Ø LOCATE 13,1:END
[UX5] 6ØØ SHELL "copy start.bat sb1.bat>nul"
[5U2] 61Ø LOCATE 13,1:END
[YV5] 7ØØ SHELL "copy basic.bat sb1.bat>nul"
[H94] 71Ø GOSUB 7ØØØ:LOCATE 13,1:END
[BN3] 8ØØ COLOR 11,1:LOCATE 7,48
[515] 81Ø PRINT KL$;STRING$(3Ø,WA$);RO$
[NH5] 82Ø LOCATE 8,79:PRINT SE$:LOCATE 9,48
[H15] 83Ø PRINT KL$;STRING$(3Ø,WA$);RU$
[PC3] 84Ø U$=BS$:L=28:Y=8:X=5Ø
[B45] 85Ø Z$=ZD$+"<>|:.,;*?\/":GOSUB 2ØØØ
[AS2] 855 LOCATE 24,1,Ø,Ø,Ø
[6A5] 86Ø IF ESC=1 THEN GOSUB 5ØØ5:GOTO 3Ø
[9A4] 87Ø COLOR 14:BS$=U$:SHELL BS$
[FB5] 88Ø PRINT:SHELL "pause":GOSUB 5ØØ5
[EN1] 89Ø GOTO 3Ø
[ES5] 9ØØ SHELL "copy ende.bat sb1.bat>nul"
[DU2] 91Ø LOCATE 14,1:END
[Z35] 95Ø IF LEN(A$)<>2 THEN BEEP:GOTO 3Ø
[D44] 96Ø A$=MID$(A$,2,1):A=Ø:TMP=IN
[C24] 97Ø IF A$="H" THEN IN=IN-1:A=1
[L24] 98Ø IF A$="P" THEN IN=IN+1:A=1
[124] 99Ø IF A=Ø THEN BEEP:GOTO 3Ø
[SH1] 995 IA=TMP
[T73] 1ØØØ IF IN=8 THEN IN=1
```

```
[783] 1Ø1Ø IF IN=Ø THEN IN=7
[HA3] 1Ø2Ø GOSUB 6ØØØ:GOTO 3Ø
[UR4] 2ØØØ A=Ø:ESC=Ø:RET=Ø:INS=Ø:ZG=1
[632] 2Ø1Ø COLOR 12,Ø
[HA5] 2Ø4Ø U$=U$+SPACE$(L-LEN(U$)):GOTO 22ØØ
[R85] 2Ø5Ø A$=INKEY$:IF A$="" THEN GOTO 2Ø5Ø
[7U1] 2Ø6Ø A=ASC(A$)
[9R3] 2Ø7Ø IF A=Ø THEN GOTO 227Ø
[5S3] 2Ø8Ø IF A=8 THEN GOTO 223Ø
[PC5] 2Ø9Ø IF A=13 THEN RET=1:A=Ø:GOTO 241Ø
[N95] 21ØØ IF A=27 THEN ESC=1:A=Ø:GOTO 241Ø
[2Z2] 211Ø JA=INSTR(Z$,A$)
[YT4] 212Ø IF JA=Ø THEN BEEP:GOTO 2Ø5Ø
[V64] 213Ø IF INS=Ø THEN GOTO 218Ø
[1W2] 214Ø U$=LEFT$(U$,L-1)
[F73] 215Ø U1$=LEFT$(U$,ZG-1)
[PY2] 216Ø U2$=MID$(U$,ZG)
[662] 217Ø U$=U1$+A$+U2$
[PY2] 218Ø MID$(U$,ZG,1)=A$
[EH2] 219Ø ZG=ZG-(ZG<L)
[FC4] 22ØØ LOCATE Y,X,Ø:PRINT U$;
[9Z3] 221Ø LOCATE Y,X+ZG-1,1,1,1Ø
[7W1] 222Ø GOTO 2Ø5Ø
[NV4] 223Ø IF ZG<2 THEN BEEP:GOTO 2Ø5Ø
[H73] 224Ø U1$=LEFT$(U$,ZG-2)
[L93] 225Ø U2$=MID$(U$,ZG)+" "
[2K4] 226Ø U$=U1$+U2$:ZG=ZG-1:GOTO 22ØØ
[SC5] 227Ø IF LEN(A$)<2 THEN BEEP:GOTO 2Ø5Ø
[514] 228Ø A=INSTR(S$,MID$(A$,2,1))
[UL4] 229Ø IF A=Ø THEN BEEP:GOTO 2Ø5Ø
[XP3] 23ØØ IF A<5 THEN GOTO 241Ø
[DS4] 231Ø ON A-4 GOTO 233Ø,234Ø,235Ø,236Ø
[8K3] 232Ø ON A-8 GOTO 239Ø,24ØØ
[L14] 233Ø ZG=ZG-(ZG<L):GOTO 221Ø
[WØ4] 234Ø ZG=ZG+(ZG>1):GOTO 221Ø
[3L3] 235Ø INS=1-INS:GOTO 2Ø5Ø
[T73] 236Ø U1$=LEFT$(U$,ZG-1)
[LH3] 237Ø U2$=MID$(U$,ZG+1)+" "
[3B3] 238Ø U$=U1$+U2$:GOTO 22ØØ
[HT2] 239Ø ZG=1:GOTO 221Ø
[6U2] 24ØØ ZG=L:GOTO 221Ø
[MØ4] 241Ø COLOR 15,Ø:LOCATE Y,X,Ø
[B12] 245Ø PRINT U$;
[YV1] 246Ø RETURN
[Y96] 3ØØØ ZG$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
[KA6] 3Ø1Ø ZK$="abcdefghijklmnopqrstuvwxyz"
[6P2] 3Ø2Ø ZZ$="123456789"
[Ø95] 3Ø3Ø ZD$=ZG$+ZK$+ZZ$+" {}~#'`!$%&()"
[L44] 3Ø4Ø S$="HPIQMKRSGO":BS$=""
[R45] 3Ø5Ø IN=1:IA=1:T$="dDeEcCsSgGbBvV"
[F94] 3Ø6Ø DATA "┌","┐","└","┘","│","─"
[ZØ4] 3Ø7Ø REM  2Ø1,187,188,2ØØ,186,2Ø5
[3T4] 3Ø8Ø READ LO$,RO$,RU$,LU$,SE$,WA$
[4U3] 3Ø9Ø DATA "╠","╣","╦","╩","╬"
[MG3] 31ØØ REM  2Ø4,185,2Ø3,2Ø2,2Ø6
[E24] 311Ø READ KL$,KR$,KO$,KU$,KX$
[7V1] 312Ø RETURN
[GW2] 4ØØØ DAT$=COMMAND$(1)
[9M5] 4Ø1Ø WK=INSTR(DAT$,"?")+INSTR(DAT$,"*")
[U15] 4Ø2Ø IF WK >Ø THEN DAT$="":GOTO 4Ø6Ø
```

```
[6C3] 4Ø3Ø PT=INSTR(DAT$,".")
[6W5] 4Ø4Ø IF PT>Ø THEN DAT$=LEFT$(DAT$,PT-1)
[GP4] 4Ø5Ø IF LEN(DAT$)>8 THEN DAT$=""
[ØP3] 4Ø6Ø IF DAT$="" THEN WX=1
[SW3] 4Ø7Ø IF WX=Ø THEN RETURN
[MZ3] 4Ø8Ø DAT$="NONAME":GOTO 35Ø
[X85] 5ØØØ FOR I=1 TO 7:READ A$(I):NEXT I
[1V2] 5ØØ5 COLOR 14,Ø:CLS
[WW3] 5Ø1Ø LOCATE 1,1:PRINT DAT$
[992] 5Ø2Ø LOCATE 1,3Ø
[HX4] 5Ø3Ø PRINT "SBasic-Compiler 1.Ø"
[U32] 5Ø4Ø COLOR 11,1
[4R6] 5Ø5Ø LOCATE 2,31:PRINT LO$;STRING$(16,WA$)RO$
[7A6] 5Ø6Ø FOR I=1 TO 7:LOCATE 2+I,31:PRINT SE$;
[A64] 5Ø7Ø PRINT A$(I);SE$:NEXT I
[5W6] 5Ø8Ø LOCATE 1Ø,31:PRINT LU$;STRING$(16,WA$)RU$
[4W1] 5Ø9Ø RETURN
[MK3] 51ØØ DATA " Dateiname      "
[3B3] 511Ø DATA " Editor         "
[SH3] 512Ø DATA " Compiler       "
[983] 513Ø DATA " Start          "
[SE3] 514Ø DATA " GW-Basic       "
[484] 515Ø DATA " Betriebssystem "
[CR3] 516Ø DATA " Verlassen      "
[X94] 6ØØØ LOCATE IA+2,32:COLOR 11,1
[KH2] 6Ø1Ø PRINT A$(IA);
[DB4] 6Ø2Ø LOCATE IN+2,32:COLOR Ø,11
[P15] 6Ø3Ø PRINT A$(IN);CHR$(29);:RETURN
[135] 7ØØØ OPEN "$$$.bas" FOR OUTPUT AS #1
[1V5] 7Ø1Ø PRINT #1,"1 CLS:KEY OFF:DEF SEG=Ø";
[2K3] 7Ø2Ø PRINT #1,":COLOR Ø,Ø"
[4H5] 7Ø3Ø PRINT #1,"2 PRINT ";CHR$(34);"cls";
[NB3] 7Ø4Ø PRINT #1,":print ";
[8S4] 7Ø5Ø PRINT #1,CHR$(34);";CHR$(34);"
[9Y4] 7Ø6Ø PRINT #1,"3 PRINT CHR$(175);";
[EW4] 7Ø7Ø PRINT #1,CHR$(34);DAT$;".BAS";
[MF3] 7Ø8Ø PRINT #1,CHR$(34);";"
[KY4] 7Ø9Ø PRINT #1,"4 PRINT CHR$(174);";
[XM5] 71ØØ PRINT #1,CHR$(34);" wird geladen";
[P94] 711Ø PRINT #1," ... ";CHR$(34);";"
[8P4] 712Ø PRINT #1,"5 PRINT CHR$(34);";
[VD4] 713Ø PRINT #1,CHR$(34);";:load";
[7F3] 714Ø PRINT #1,CHR$(34);";"
[NP4] 715Ø PRINT #1,"6 PRINT CHR$(34);";
[3U3] 716Ø PRINT #1,CHR$(34);DAT$
[U15] 717Ø PRINT #1,"7 POKE 1Ø5Ø,3Ø:POKE ";
[913] 718Ø PRINT #1,"1Ø52,34"
[AY4] 719Ø PRINT #1,"8 POKE 1Ø54,Ø:POKE ";
[RØ3] 72ØØ PRINT #1,"1Ø55,71"
[X15] 721Ø PRINT #1,"9 POKE 1Ø56,13:POKE ";
[P13] 722Ø PRINT #1,"1Ø58,28"
[K65] 723Ø PRINT #1,"1Ø LOCATE 2:COLOR 7,Ø"
[W23] 724Ø CLOSE 1:RETURN
```

Listing 1. »sbshell.bas« koordiniert die neue Entwicklungsumgebung

```
[HN4] 1 'Programmname: preproz.bas
[8M6] 2 '(c) 1989 Redaktionsbüro Everts&Hagedorn
[M63] 3 'DOS International
[934] 4 'Autor: Dieter Bäumler
[6BØ] 5 '
[BZ3] 6 DIM ZLNR(1ØØØ),LABEL$(1ØØØ)
[A52] 7 Z$="1234567890"
[K46] 8 GR$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"+Z$
[Y46] 9 KL$="abcdefghijklmnopqrstuvwxyz"+Z$
[1P2] 1Ø P2$=GR$+KL$+"_."+Z$
[952] 11 FOR I%=1 TO 32
[N91] 12 READ A$
[7F2] 13 PRF$=PRF$+" "+A$
[FE1] 14 NEXT I%
[D15] 15 PRF$=PRF$+SPACE$(255-LEN(PRF$))
[ET2] 16 DAT$=COMMAND$(1)
[675] 17 OPEN DAT$+".asc" FOR INPUT AS #1
[W45] 18 OPEN DAT$+".$$$" FOR OUTPUT AS #2
[PW2] 19 WHILE NOT EOF(1)
[D83] 2Ø LINE INPUT #1,ZEIL$
[LM1] 21 GOSUB 43
[CY3] 22 IF ZEIL$="" THEN ZEIL$=":"
[UE2] 23 ZEILE=ZEILE+1
[3N1] 24 GOSUB 54
[3N1] 25 GOSUB 81
[PU3] 26 IF WORT%=Ø THEN GOTO 3Ø
[9P4] 27 IF LEN(WORT$)>32 THEN GOTO 3Ø
[FN1] 28 GOSUB 91
[ZU3] 29 IF WORT%=Ø THEN GOTO 3Ø
[9A5] 3Ø REM Nummer und Zeile speichern
[UN1] 31 GOSUB 116
[FD2] 32 PRINT #2,ZEIL$
[U21] 33 WEND
[ED1] 34 CLOSE 1
[MD1] 35 CLOSE 2
[Ø24] 36 IF ZL%<=1ØØØ THEN GOTO 13Ø
[TE3] 37 REM Zu viele Marken
[UD1] 38 CLOSE 1
[ØE1] 39 CLOSE 2
```

```
[WG2] 4Ø KILL DAT$+".$$$"
[SØ5] 41 BEEP:PRINT "Zu viele Marken !"
[BTØ] 42 END
[F64] 43 REM Leerzeichen vorne ...
[ZW3] 44 IF ZEIL$="" THEN GOTO 52
[YP4] 45 IF ASC(ZEIL$)<>32 THEN GOTO 48
[473] 46 ZEIL$=MID$(ZEIL$,2)
[BE1] 47 GOTO 43
[KV4] 48 REM ... und hinten abschneiden
[5R5] 49 IF RIGHT$(ZEIL$,1)<>" " THEN GOTO 52
[RØ5] 5Ø ZEIL$=LEFT$(ZEIL$,LEN(ZEIL$)-1)
[SX3] 51 IF ZEIL$<>"" THEN GOTO 48
[GX1] 52 REM Fertig
[XC1] 53 RETURN
[FY5] 54 REM In Großbuchstaben umwandeln
[BC4] 55 KLEIN$=MID$(KL$,1,26):MRK%=Ø
[SC2] 56 LL%=LEN(ZEIL$)
[VG3] 57 LAST$="XXXX"+CHR$(Ø)
[2C2] 58 FOR I%=1 TO LL%
[9S4] 59 BEM%=SGN(INSTR(LAST$,"REM "))
[ES3] 6Ø IF BEM%=1 THEN GOSUB 75
[EN4] 61 IF BEM%=1 THEN I%=LL%:GOTO 73
[TK3] 62 REM2%=INSTR(LAST$,"'")
[KU4] 63 IF REM2%<>Ø THEN I%=LL%:GOTO 73
[LY2] 64 X$=MID$(ZEIL$,I%,1)
[ES4] 65 IF X$=CHR$(34) THEN MRK%=1-MRK%
[DH3] 66 IF MRK%=1 THEN GOTO 73
[S83] 67 X%=INSTR(KLEIN$,X$)
[6D3] 68 IF X%=Ø THEN GOTO 71
[TG2] 69 X$=MID$(GR$,X%,1)
[HY2] 7Ø MID$(ZEIL$,I%,1)=X$
[9E6] 71 REM letzten fünf Zeichen festhalten
[2K3] 72 LAST$=MID$(LAST$,2)+X$
[SE1] 73 NEXT I%
[4D1] 74 RETURN
[LN3] 75 REM Tatsächlich REM ?
[BZ2] 76 WERT%=ASC(LAST$)
[BY3] 77 IF WERT%=Ø  THEN RETURN
[AØ4] 78 IF WERT%=32 THEN RETURN
[314] 79 IF WERT%=58 THEN RETURN
[WE2] 8Ø BEM%=Ø:RETURN
[ØZ3] 81 REM Makre ausschneiden
[HT3] 82 WORT%=INSTR(ZEIL$,":")
[ØU3] 83 IF WORT%=Ø THEN RETURN
[7E5] 84 IF WORT%=LEN(ZEIL$) THEN GOTO 88
[E64] 85 TMP$=MID$(ZEIL$,WORT%+1,1)
[TW3] 86 IF TMP$=" " THEN GOTO 88
[1S2] 87 WORT%=Ø:RETURN
[8Y1] 88 REM Fertig
[KD4] 89 WORT$=LEFT$(ZEIL$,WORT%-1)
[WC1] 9Ø RETURN
[DL5] 91 REM Handelt es sich um eine Marke ?
[PF2] 92 LW%=LEN(WORT$)
[ØM3] 93 IF LW%=Ø THEN RETURN
[172] 94 FOR I=1 TO LW%
[7X2] 95 A$=MID$(WORT$,I,1)
[YP2] 96 A%=INSTR(P2$,A$)
[9A4] 97 IF A%=Ø THEN I=LEN(WORT$)+1
[WB1] 98 NEXT I
[8L4] 99 IF A%=Ø THEN WORT%=Ø:RETURN
[1S2] 1ØØ LW%=LEN(WORT$)
[1N3] 1Ø1 IF LW%>9 THEN GOTO 1Ø5
[X55] 1Ø2 TEST%=INSTR(PRF$," "+WORT$+" ")
[544] 1Ø3 IF TEST%=Ø THEN GOTO 1Ø5
[UX2] 1Ø4 WORT%=Ø:RETURN
[D43] 1Ø5 REM Marke ist ok
[CS1] 1Ø6 ZL%=ZL%+1
[MX3] 1Ø7 IF ZL%>1ØØØ THEN GOTO 37
[783] 1Ø8 ZLNR(ZL%)=ZEILE
[3Ø3] 1Ø9 LABEL$(ZL%)=WORT$
[WH5] 11Ø IF WORT%=LEN(ZEIL$) THEN GOTO 113
[9S5] 111 ZEIL$=MID$(ZEIL$,WORT%+2)+" '"+WORT$
[3M1] 112 RETURN
[BA4] 113 REM Marke steht allein
[1R3] 114 ZEIL$="REM *"+WORT$+"*"
[FM1] 115 RETURN
[ZP5] 116 REM Nummer zu String umwandeln
[P44] 117 NR$=MID$(STR$(ZEILE),2)
[UA3] 118 ZEIL$=NR$+" "+ZEIL$
[ZM1] 119 RETURN
[G76] 12Ø REM Das dürfen keine Marken sein:
[7H5] 121 DATA "RESTORE","LPRINT","PRINT"
[5L5] 122 DATA "SYSTEM","BEEP","RANDOMIZE"
[HA5] 123 DATA "REM","MOTOR","RESET","END"
[1R5] 124 DATA "TROFF","NEW","RESUME","TRON"
[PP5] 125 DATA "NEXT","RETURN","CLEAR","CLS"
[395] 126 DATA "RUN","SHELL","WEND","LIST"
[NØ6] 127 DATA "LLIST","FILES","WRITE","DATA"
[VN5] 128 DATA "STOP","ELSE","RETURN","GOTO"
[N83] 129 DATA "GOSUB","RUN"
[E74] 13Ø REM Zweiter Durchgang
[5W4] 131 OPEN DAT$+".$$$" FOR INPUT AS #1
[HL5] 132 OPEN DAT$+".bas" FOR OUTPUT AS #2
[AX2] 133 WHILE NOT EOF(1)
[8S3] 134 LINE INPUT #1,ZEILE$
[DU1] 135 GOSUB 143
```

Listing 2. Der Präprozessor »preproz.bas« wandelt strukturierten Quelltext in das GW-Basic-Format um

```
[1H4] 136 PRINT #2,ZLNR$;ZEILE$;BEM$
[G71] 137 WEND
[2N1] 138 CLOSE 1
[AN1] 139 CLOSE 2
[WT1] 140 GOSUB 320
[LU1] 141 GOSUB 339
[A31] 142 END
[HA5] 143 REM Programmzeile behandeln
[YN3] 144 B1%=1:B2%=1:GT%=1:GS%=1
[U83] 145 RS%=1:TH%=1:BEM$=""
[YM3] 146 NR%=INSTR(ZEILE$," ")
[E34] 147 ZLNR$=LEFT$(ZEILE$,NR%)
[734] 148 ZEILE$=MID$(ZEILE$,NR%+1)
[1K3] 149 LAENGE%=LEN(ZEILE$)
[6B4] 150 IF LAENGE%<5 THEN RETURN
[4Z3] 151 REM Auf REM prüfen
[UU3] 152 MRK%=0:LL%=LEN(ZEILE$)
[ZU4] 153 TMP%=0:LAST$="XXXX"+CHR$(0)
[0X2] 154 FOR I%=1 TO LL%
[NZ4] 155 BEM%=SGN(INSTR(LAST$,"REM "))
[WY3] 156 IF BEM%=1 THEN GOSUB 75
[MY3] 157 IF BEM%=1 THEN GOTO 168
[714] 158 BEM%=INSTR(LAST$,"'")-3
[VY3] 159 IF BEM%>0 THEN GOTO 168
[M73] 160 X$=MID$(ZEILE$,I%,1)
[GY4] 161 IF X$=CHR$(34) THEN MRK%=1-MRK%
[KY3] 162 IF MRK%=1 THEN GOTO 164
[FS3] 163 LAST$=MID$(LAST$,2)+X$
[8U2] 164 NEXT I% 'WTREM
[RH3] 165 LAENGE%=LEN(ZEILE$)
[2C4] 166 IF LAENGE%<6 THEN RETURN
[WR1] 167 GOTO 173
[5T4] 168 REM Bemerkung abschneiden
[725] 169 IF BEM%=2 THEN BEM%=1 ELSE BEM%=4
[8B4] 170 BEM$=MID$(ZEILE$,I%-BEM%)
[8Y4] 171 ZEILE$=LEFT$(ZEILE$,I%-BEM%-1)
[GT2] 172 I%=LL%:GOTO 164
[K14] 173 REM Auf GOTO prüfen
[NY4] 174 GT%=INSTR(GT%,ZEILE$,"GOTO ")
[3P3] 175 IF GT%=0 THEN GOTO 196
[4X2] 176 U%=GT%:GOSUB 260
[R95] 177 IF U%=0 THEN GT%=GT%+4:GOTO 173
[HV1] 178 GOSUB 266
[Z95] 179 IF U%=0 THEN GT%=GT%+4:GOTO 173
[U13] 180 U%=GT%+5:GOSUB 274
[YN5] 181 IF LABEL$="" THEN GT%=GT%+4:GOTO 173
[DG3] 182 U$=LABEL$:GOSUB 298
[B84] 183 IF LBLNR$="" THEN GOTO 350
[8V1] 184 GOSUB 309
[SB4] 185 REM Kommt noch ",Marke" ?
[W45] 186 IF LEN(ZEILE2$)<2 THEN GOTO 196
[4Z4] 187 WT%=INSTR(U%,ZEILE$,ZEILE2$)
[FU3] 188 WT$=MID$(ZEILE$,WT%,1)
[NS5] 189 IF WT$<>"," THEN GT%=GT%+4:GOTO 173
[923] 190 U%=WT%+1:GOSUB 274
[1P5] 191 IF LABEL$="" THEN GT%=GT%+4:GOTO 173
[6V1] 192 GOSUB 298
[E84] 193 IF LBLNR$="" THEN GOTO 350
[BV1] 194 GOSUB 309
[3S1] 195 GOTO 185
[Z34] 196 REM Auf GOSUB prüfen
[GC4] 197 IF LAENGE%<7 THEN RETURN
[A75] 198 GS%=INSTR(GS%,ZEILE$,"GOSUB ")
[SN3] 199 IF GS%=0 THEN GOTO 220
[RV2] 200 U%=GS%:GOSUB 260
[T85] 201 IF U%=0 THEN GS%=GS%+5:GOTO 196
[6U1] 202 GOSUB 266
[195] 203 IF U%=0 THEN GS%=GS%+5:GOTO 196
[L13] 204 U%=GS%+6:GOSUB 274
[5P5] 205 IF LABEL$="" THEN GS%=GS%+5:GOTO 196
[7G3] 206 U$=LABEL$:GOSUB 298
[584] 207 IF LBLNR$="" THEN GOTO 350
[2V1] 208 GOSUB 309
[KB4] 209 REM Kommt noch ",Marke" ?
[625] 210 IF LEN(ZEILE2$)<2 THEN GOTO 220
[TX4] 211 WT%=INSTR(U%,ZEILE$,ZEILE2$)
[4T3] 212 WT$=MID$(ZEILE$,WT%,1)
[RR5] 213 IF WT$<>"," THEN GS%=GS%+5:GOTO 196
[323] 214 U%=WT%+1:GOSUB 274
[8P5] 215 IF LABEL$="" THEN GS%=GS%+5:GOTO 196
[0V1] 216 GOSUB 298
[884] 217 IF LBLNR$="" THEN GOTO 350
[5V1] 218 GOSUB 309
[4R1] 219 GOTO 209
[DR4] 220 REM Auf RESTORE prüfen
[HA4] 221 IF LAENGE%<9 THEN GOTO 234
[2L5] 222 RS%=INSTR(RS%,ZEILE$,"RESTORE ")
[6N3] 223 IF RS%=0 THEN GOTO 234
[SW2] 224 U%=RS%:GOSUB 260
[EF3] 225 IF U%=0 THEN GOTO 233
[WU1] 226 GOSUB 266
[PF3] 227 IF U%=0 THEN GOTO 233
[P23] 228 U%=RS%+8:GOSUB 274
[D64] 229 IF LABEL$="" THEN GOTO 233
[SF3] 230 U$=LABEL$:GOSUB 298
[P74] 231 IF LBLNR$="" THEN GOTO 350
[LU1] 232 GOSUB 309
[N33] 233 RS%=RS%+7:GOTO 220
[MZ3] 234 REM Auf THEN prüfen
[LB4] 235 IF LAENGE%<11 THEN RETURN
[XZ4] 236 TH%=INSTR(TH%,ZEILE$,"THEN ")
[VN3] 237 IF TH%=0 THEN RETURN
[RW2] 238 U%=TH%:GOSUB 260
[EG3] 239 IF U%=0 THEN GOTO 249
[AU1] 240 GOSUB 266
[HF3] 241 IF U%=0 THEN GOTO 249
[D13] 242 U%=TH%+5:GOSUB 274
[F64] 243 IF LABEL$="" THEN GOTO 249
[DU1] 244 GOSUB 250
[R64] 245 IF LABEL$="" THEN GOTO 249
[LG3] 246 U$=LABEL$:GOSUB 298
[994] 247 IF LBLNR$="" THEN GOTO 249
[EV1] 248 GOSUB 309
[914] 249 TH%=TH%+4:GOTO 234'THEND
[KB6] 250 REM Wirklich eine Marke nach THEN ?
[D03] 251 X%=U%+LEN(LABEL$)
[5L4] 252 IF X%>=LAENGE% THEN RETURN
[S03] 253 X$=MID$(ZEILE$,X%)
[CN2] 254 X2$=LEFT$(X$,1)
[4T3] 255 IF X2$=":" THEN RETURN
[395] 256 IF X2$<>"E" THEN LABEL$="":RETURN
[RE3] 257 X%=INSTR(X$,"ELSE ")
[3W3] 258 IF X%<>1 THEN LABEL$=""
[DN1] 259 RETURN
[UH5] 260 REM Ist der Befehl ein Befehl ?
[0L3] 261 IF U%=1 THEN RETURN
[ZD3] 262 A$=MID$(ZEILE$,U%-1,1)
[8G3] 263 IF A$=" " THEN RETURN
[6H3] 264 IF A$=":" THEN RETURN
[1F2] 265 U%=0:RETURN
[FG4] 266 REM Ist der Befehl in "" ?
[LN1] 267 MARK%=0
[YN2] 268 FOR I%=1 TO U%
[C73] 269 A$=MID$(ZEILE$,I%,1)
[E65] 270 IF A$=CHR$(34) THEN MARK%=1-MARK%
[7H1] 271 NEXT I
[CH2] 272 U%=U%*(1-MARK%)
[VM1] 273 RETURN
[853] 274 REM Marke holen
[HX1] 275 LABEL$="":
[G55] 276 IF U%>LAENGE% THEN U%=0:RETURN
[DZ1] 277 X%=U%:LK=1
[1B2] 278 REM *KLSL3*
[Y73] 279 A$=MID$(ZEILE$,X%,1)
[N73] 280 IF A$<>" " THEN LK=0
[YH3] 281 IF LK=1 THEN GOTO 284
[TU2] 282 A%=INSTR(P2$,A$)
[HF3] 283 IF A%=0 THEN GOTO 287
[1W2] 284 LABEL$=LABEL$+A$
[MB5] 285 IF X%=LAENGE% THEN A$=":":GOTO 287
[WT2] 286 X%=X%+1:GOTO 278
[D92] 287 REM Fertig
[WX4] 288 IF A$=":" OR A$="," THEN RETURN
[WS3] 289 IF A$=" " THEN GOTO 291
[D33] 290 LABEL$="":RETURN
[7Y5] 291 REM Leerzeichen danach mitlesen
[0U2] 292 LABEL$=LABEL$+" "
[EE4] 293 IF X%=LAENGE% THEN RETURN
[TF1] 294 X%=X%+1
[M73] 295 A$=MID$(ZEILE$,X%,1)
[LS3] 296 IF A$=" " THEN GOTO 291
[HN1] 297 RETURN
[5E4] 298 REM Zeilennummer holen
[961] 299 NR=0
[VY2] 300 FOR I%=1 TO ZL%:
[6A4] 301 PRFLABEL$=" "+LABEL$(I%)+" "
[CL5] 302 LL%=INSTR(" "+LABEL$+" ",PRFLABEL$)
[174] 303 IF LL%>0 THEN NR=I%:I%=ZL%
[MN1] 304 NEXT I%
[SD4] 305 IF NR>0 THEN NR=ZLNR(NR)
[0Y4] 306 IF NR=0 THEN LBLNR$="":RETURN
[044] 307 LBLNR$=MID$(STR$(NR),2)
[WM1] 308 RETURN
[EY5] 309 REM Marke durch Nummer ersetzen
[TC4] 310 ZEILE1$=LEFT$(ZEILE$,U%-1)
[4P5] 311 ZEILE2$=MID$(ZEILE$,U%+LEN(LABEL$))
[XG4] 312 IF ZEILE2$="" THEN GOTO 316
[0C3] 313 A$=LEFT$(ZEILE2$,1)
[BU4] 314 IF A$="," OR A$=":" THEN GOTO 316
[763] 315 ZEILE2$=" "+ZEILE2$
[RN2] 316 REM Passiert
[S15] 317 ZEILE$=ZEILE1$+LBLNR$+ZEILE2$
[SH3] 318 LAENGE%=LEN(ZEILE$)
[3N1] 319 RETURN
[PP3] 320 REM $$$.bas erzeugen
[375] 321 PZ1$="10 KEY OFF:CLS:DEF SEG=0"
[GN3] 322 PZ1$=PZ1$+":COLOR 0,0"
[SS4] 323 PZ2$="20 KEY 1,"+CHR$(34)+"load"
[7R4] 324 PZ2$=PZ2$+CHR$(34)+"+CHR$(34)+"
[FV3] 325 PZ2$=PZ2$+CHR$(34)+DAT$
[3K4] 326 PZ2$=PZ2$+CHR$(34)+"+CHR$(13)"
[1V4] 327 PZ3$="30 KEY 2,"+CHR$(34)+"save"
[WR4] 328 PZ3$=PZ3$+CHR$(34)+"+CHR$(34)+"
[4W3] 329 PZ3$=PZ3$+CHR$(34)+DAT$
[LH4] 330 PZ3$=PZ3$+CHR$(34)+"+CHR$(13)"
[X65] 331 PZ4$="40 KEY 3,"+CHR$(34)+"system"
```

```
[ØK4] 332 PZ4$=PZ4$+CHR$(34)+"+CHR$(13)"
[X85] 333 PZ5$="5Ø POKE 1Ø5Ø,3Ø:POKE 1Ø52,36"
[XF5] 334 PZ6$="6Ø FOR I=1Ø54 TO 1Ø59 STEP 2"
[DL3] 335 PZ6$=PZ6$+":POKE I,Ø"
[YR4] 336 PZ7$="7Ø POKE I+1,(I-1Ø54)/2+59"
[ZF3] 337 PZ7$=PZ7$+":NEXT I"
[5N1] 338 RETURN
[2U3] 339 REM $$$.bas speichern
[3W4] 34Ø OPEN "$$$.bas" FOR OUTPUT AS #1
[WE2] 341 PRINT #1,PZ1$
[1F2] 342 PRINT #1,PZ2$
[6F2] 343 PRINT #1,PZ3$
[BF2] 344 PRINT #1,PZ4$
[GF2] 345 PRINT #1,PZ5$
[NF2] 346 PRINT #1,PZ6$
[UF2] 347 PRINT #1,PZ7$
[9N1] 348 CLOSE 1
[CN1] 349 RETURN
[TW3] 35Ø REM Unbekannte Marke
[294] 351 PRINT ZLNR$;ZEILE$;BEM$
[XC1] 352 PRINT
[RM5] 353 PRINT "Unbekannte Marke: ";LABEL$
[BV1] 354 BEEP:END
```

Listing 2. Der Präprozessor »preproz.bas« wandelt strukturierten Quelltext in das GW-Basic-Format um

```
[4U2]    1: gosub bildschirm
[X42]    2: gosub abfrage
[3V2]    3: gosub korrektur
[SF4]    4: if ok=1 then gosub speichern
[MB2]    5: gosub abschied
[4MØ]    6: end
[ØØØ]    7:
[YØ2]    8: bildschirm:
[762]    9: cls:color Ø,11
[W15]   1Ø: print "Batchdateien-Generator"
[PG2]   11: color 11,Ø:print
[23E]   12: print "Dieses Programm generiert die von SB
             benötigten Stapeldateien mit der Endung"
[42D]   13: print "»bat«. Geben Sie im Folgenden bitte
            die Namen der Programme an, die man von"
[1GD]   14: print "Ihnen erfragt. Nur dann, wenn sich d
            iese Programme im gleichen Verzeichnis"
[FZD]   15: print "befinden, in dem auch »sb.bat« vorha
            nden ist, oder wenn die entsprechenden"
[ZBE]   16: print "Verzeichnisse in der Umgebungsvariab
            len »path« aufgeführt sind, können Sie"
[HX9]   17: print "auf die Pfadangabe für die jeweilige
             Datei verzichten."
[WC3]   18: print:print:return
[ØØØ]   19:
[RA1]   2Ø: abfrage:
[PU1]   21: for i=1 to 3
[6W5]   22:   on i gosub frag_ed,frag_bas,frag_sb
[U11]   23: next i
[W52]   24: print:print
[651]   25: return
[ØØØ]   26:
[4D1]   27: frag_ed:
[3X4]   28: color 12:print "Editor           : ";
[NG4]   29: color 14:line input editor$
[651]   3Ø: return
[ØØØ]   31:
[WG1]   32: frag_bas:
[RZ5]   33: color 12:print "Basic-Interpreter: ";
[GN4]   34: color 14:line input gwbasic$
[651]   35: return
[ØØØ]   36:
[6E1]   37: frag_sb:
[LS5]   38: color 12:print "SBasic-Compiler  : ";
[HE4]   39: color 14:line input sbasic$
[651]   4Ø: return
[ØØØ]   41:
[ZV1]   42: korrektur:
[ZØ3]   43: color 12:print "1. ";
[9E3]   44: print "Editor          :";
[KK3]   45: color 14:print editor$
[113]   46: color 12:print "2. ";
[1K4]   47: print "Basic-Interpreter:";
[SR3]   48: color 14:print gwbasic$
[313]   49: color 12:print "3. ";
[F84]   5Ø: print "SBasic-Compiler  :";
[WG3]   51: color 14:print sbasic$
[GXØ]   52: print
[B71]   53: color 11
[569]   54: print "Drücken Sie bitte die Taste [Ø], um
            zu speichern,"
[XZ7]   55: print "[1] bis [3], um eine Angabe zu korri
            gieren,"
[BT7]   56: print "oder [Esc], um das Programm abzubrec
            hen."
[GXØ]   57: print
[DN1]   58: taste: a$=""
[GL1]   59: while a$=""
[XA1]   6Ø:   a$=inkey$
[TRØ]   61: wend
[CU3]   62: if a$="Ø" then ok=1:return
[VS4]   63: if a$=chr$(27) then ok=Ø:return
[WA1]   64: a=val(a$)
[G65]   65: if a<1 or a>3 then beep:goto taste
[GXØ]   66: print
[ZW5]   67: on a gosub frag_ed, frag_bas, frag_sb
[2M3]   68: print:goto korrektur
[ØØØ]   69:
[1S1]   7Ø: speichern:
[RA5]   71: open "basic.bat" for output as #1
[1C3]   72:   print #1,gwbasic$;" $$$"
[6V2]   73:   anz=2:gosub zlhs
[751]   74: close 1
[ØØØ]   75:
[RX4]   76: open "ende.bat" for output as #1
[4V2]   77:   anz=1:gosub zlhs
[751]   78: close 1
[ØØØ]   79:
[3Z4]   8Ø: open "comp.bat" for output as #1
[AV2]   81:   anz=4:gosub zlhs
[1C3]   82:   print #1,gwbasic$;" $$$"
[6V2]   83:   anz=2:gosub zlhs
[133]   84:   print #1,sbasic$;" %1"
[GV2]   85:   anz=7:gosub zlhs
[751]   86: close 1
[ØØØ]   87:
[4Z4]   88: open "edit.bat" for output as #1
[YR3]   89:   print #1,editor$;" %1.asc"
[4V2]   9Ø:   anz=1:gosub zlhs
[751]   91: close 1
[ØØØ]   92:
[9F5]   93: open "start.bat" for output as #1
[MV2]   94:   anz=9:gosub zlhs
[751]   95: close 1
[ØØØ]   96:
[651]   97: return
[ØØØ]   98:
[ZVØ]   99: zlhs:
[E82]  1ØØ: for i=1 to anz
[ØS2]  1Ø1:   read a$:print #1,a$
[U11]  1Ø2: next i
[651]  1Ø3: return
[ØØØ]  1Ø4:
[GØ1]  1Ø5: basic:
[AD2]  1Ø6: DATA "del $$$.bas"
[5N1]  1Ø7: DATA "sb %1"
[XTØ]  1Ø8: ende:
[V54]  1Ø9: DATA "echo SBasic beendet."
[BVØ]  11Ø: comp:
[584]  111: DATA "if exist %1.$$$ del %1.$$$"
[NV5]  112: DATA "if not exist %1.asc goto fehler"
[2K2]  113: DATA "preproz %1"
[RM5]  114: DATA "if not exist $$$.bas goto ende"
[AD2]  115: DATA "del $$$.bas"
[M22]  116: DATA "del %1.$$$"
[5F2]  117: DATA "goto ende"
[H42]  118: DATA ":fehler"
[BE6]  119: DATA "echo Datei %1.asc existiert nicht!"
[DU1]  12Ø: DATA ":ende"
[GT1]  121: DATA "echo."
[WY1]  122: DATA "pause"
[5N1]  123: DATA "sb %1"
[2UØ]  124: edit:
[5N1]  125: DATA "sb %1"
[E21]  126: start:
[EW5]  127: DATA "if not exist %1.exe goto fehler"
[B41]  128: DATA "%1"
[5F2]  129: DATA "goto ende"
[H42]  13Ø: DATA ":fehler"
[AF6]  131: DATA "echo Datei %1.exe existiert nicht!"
[DU1]  132: DATA ":ende"
[GT1]  133: DATA "echo."
[WY1]  134: DATA "pause"
[5N1]  135: DATA "sb %1"
[ØØØ]  136:
[EG1]  137: abschied:
[GXØ]  138: print
[ZK5]  139: if ok=1 then print "Gespeichert..."
[KG5]  14Ø: if ok=Ø then print "Abgebrochen..."
[651]  141: return
```

Listing 3. Das Demonstrationsprogramm »makebat.asc« erzeugt die für »sbshell« ablaufwichtigen Stapeldateien

```
[DA1]    1: echo off
[VM1]    2: sbshell %1
[DTØ]    3: sb1 %1
```

Listing 4. »sb.bat« ist das erste Glied der Kette sich gegenseitig aufrufender Stapeldateien

Maschinen-sprache inklusive

Das Programmieren in einer Hochsprache ist eine feine Sache. Leider gibt es jedoch einige Probleme, die sich aufgrund extremen Zeitmangels nur in Maschinensprache lösen lassen. Der folgende Beitrag zeigt Ihnen, wie Sie Assemblerroutinen mit Ihren Programmen verknüpfen.

Fast alle Hochsprachen außer dem GW-Basic-Interpreter erlauben es, Maschinenprogramme oder andere Daten als Obj-Dateien einzubinden. Die Schwierigkeit dabei ist zum einen, daß Sie einen Assembler besitzen müssen, da sich mit Debug keine Obj-Dateien erzeugen lassen. Zum anderen möchten Sie sicher auch einmal Daten in ein Programm aufnehmen, die nicht als Assemblerprogramm vorliegen. Wer hier nicht auf Borlands »binobj.exe« zurückgreifen kann, dürfte ebenfalls beträchtliche Schwierigkeiten bekommen. Und schließlich gibt es noch das gute alte GW-Basic, das Maschinenprogramme nur in Form von Data-Zeilen verdaut.

Für alle diese Fälle ist das Utility »datgen.pas« vorgesehen. Geben Sie die Listings 1 und 2 ein, und compilieren Sie sie mit Turbo Pascal 4.0 oder einer höheren Version. Falls Sie übrigens das Programm »emstest.pas« (DOS International 4'89) abgetippt haben, so besitzen Sie Listing 2 bereits und brauchen es nicht noch einmal einzugeben. Jetzt rufen Sie »Datgen« wie folgt auf:

```
datgen eingabe [ausgabe] [optionen]
```

Dabei ist »eingabe« diejenige Datei, die umgewandelt werden soll. Wird der Name der Ausgabedatei nicht angegeben, so verwendet Datgen den Namen der Eingabedatei und hängt je nach Programmiersprache die Erweiterung »asm«, »bas« oder »pas« an. Folgende Optionen stehen Ihnen zur Verfügung:

»/a« legt das Assembler-Format fest. Eine Zeile besteht jeweils aus vier Leerzeichen und den Befehlen »db« beziehungsweise »dw«. Danach schließen sich die Daten an, wobei Hexadezimalzahlen, die mit einem Buchstaben beginnen, eine Null vorangesetzt bekommen.

»/b« legt das Basic-Format fest. Diese Option braucht nicht mit angegeben zu werden, es ist die Standardeinstellung. Eine Zeile besteht aus dem Befehlswort »data« und, mit Kommata getrennt, den Werten.

»/p« legt das Pascal-Format fest. Hier wird einer Zeile nichts vorangestellt, den Befehl »inline« muß man also später selbst im Editor hinzufügen. Die Daten sind jeweils durch einen Schrägstrich getrennt.

»/o« (overwrite) schaltet die Sicherheitsabfrage vor dem Überschreiben der Datei ab. Diese Abfrage soll verhindern, daß aus Versehen das gleichnamige Hochsprachenprogramm überschrieben wird.

»/d« gibt die Daten in dezimaler Form aus. Die Standardeinstellung ist hexadezimal.

»/w« gibt die Daten als Worte aus (normalerweise Byte-Ausgabe). Hat die Eingabedatei eine ungerade Länge, dann wird ein Null-Byte hinzugefügt.

»/l[#]« legt die maximale Länge einer Datenzeile fest. Als Werte sind hier Zahlen zwischen 10 und 255 zulässig, die Standardeinstellung ist 79. Datgen überprüft bei jedem neuen Datum, ob es noch in die Zeile paßt. Wenn nicht, dann gibt es die Zeile aus und beginnt eine neue.

»/n[#]« stellt die Zeilenende-Erkennung auf eine feste Anzahl an Daten um. Die Länge einer Zeile spielt nun keine Rolle mehr. Zulässige Werte liegen zwischen 1 und 255, vorgegeben ist der Wert 16.

»/z[#[,#]]« numeriert die Zeilen schrittweise durch. Die erste Zahl gibt die Anfangszeile, die zweite die Schrittweite an. Die Schrittweite darf nicht 0 sein, Vorgaben sind 100 und 1. Die Zeilennummer wird bei der Längenberechnung berücksichtigt.

Alle diese Optionen können Sie in beliebiger Reihenfolge verwenden, die Trennung mit Leerzeichen ist nicht nötig. Bei sich gegenseitig ausschließenden Optionen (etwa »/a« und »/b«) ist die jeweils letzte ausschlaggebend. Wenn Ihnen die Standardvorgaben nicht gefallen, dann ändern Sie diese doch einfach in der Konstanten-Definition.

Beim GW-Basic-Interpreter taucht ein zusätzliches Problem auf: Wo bringt man die Maschinenprogramme unter? Es gibt genau drei Lösungen, die jeweils ihre Vor- und Nachteile haben. Die erste bietet der Befehl »clear« an. Wie Sie vielleicht wissen, sind alle Daten von GW-Basic (außer dem Maschinencode des Interpreters) in einem einzigen Segment untergebracht, das normalerweise 64 KByte groß ist. Mit dem Clear-Befehl läßt sich nun die Größe dieses Segments verringern und somit vom oberen Teil ein Stück für Maschinenprogramme abzweigen. Nach der Anweisung

```
clear ,64999
```

zum Beispiel ist der Speicher von Adresse 65000 bis 65535 für Basic tabu und läßt sich deshalb für eigene Zwecke verwenden. Diese Methode hat nur den Nachteil, daß sie nicht vom Compiler SBasic (aus dem Basic-Sonderheft) unterstützt wird.

Gesucht ist eine Endlagerstätte für den Maschinencode

Die zweite und wohl am häufigsten verwendete Lösung ist die Unterbringung von Maschinenprogrammen in Strings. Die Adresse der Daten erhalten Sie dabei durch die Befehlsfolge

```
peek(varptr(mc$)+1) + peek(varptr(mc$)+2)*256
```

Werden die Strings am Anfang eines Programms definiert, so braucht man diese Adresse nur einmal zu berechnen, weil die »Müllabfuhr« (garbage collection) hier nichts umstellt. Der Nachteil dabei liegt in der Natur der Strings: Sie dürfen maximal 255 Zeichen groß sein. Größere Maschinenprogramme muß man also aufspalten, wobei umständliche Adreßberechnungen anfallen.

Die dritte und letzte Methode liest die Daten in ein eindimensionales Integer-Feld ein. Hier ergibt sich zudem der Vorteil, daß Sie recht einfach auf bestimmte Parameter zugreifen können und nicht endlos lange »Call-Würmer« produzieren (siehe »Interrupts für GW-Basic«, DOS International 12'88, Seite 72). Allerdings verschiebt der Interpreter bei jedem Anlegen einer neuen Variablen die Felder nach oben, so daß sich die Adresse des Maschinenprogramms ständig ändert. Deshalb sollte sie vor jedem Call-Befehl mit »varptr(mc%(0))« neu berechnet werden, damit der Aufruf nicht ins Leere geht.

Auch bei Parametern, mit denen ein Maschinenprogramm arbeiten soll, ist einiges zu beachten. Es ist nicht möglich, Konstanten, Rechen- oder Funktionsergebnisse (»call by value«) zu übergeben. Nur Zeiger auf Variablen (»call by reference«) kann der Interpreter verarbeiten. Diese belegen jeweils zwei Byte auf dem Stack, wobei der übergebene Wert exakt dem Funktionsergebnis von »varptr« entspricht. Zusätzlich zu den Parametern kommt noch die Rücksprungadresse, die als Far-Wert vier Byte in Anspruch nimmt. Nach dem Aufruf eines Maschinenprogramms mit

```
call mc(a$, b%)
```

sieht also der Stack folgenermaßen aus:

```
[SP+0] = Rücksprungadresse
[SP+4] = varptr(B%)
[SP+6] = varptr(A$)
```

Zu beachten ist dabei, daß die Parameter genau in der umgekehrten Reihenfolge auftreten, wie sie in der Call-Anweisung festgelegt sind. Zweckmäßigerweise ordnet man den Wert des Stack Pointers dem Base Pointer zu, weil es keine SP-relative Adressierung beim 8086 gibt. Nach dem Abarbeiten des Maschinenprogramms sind schließlich noch zwei Dinge für den Rücksprung zu beachten: Zum einen müssen Sie den Assembler dazu veranlassen, ein Intersegment-Return zu erzeugen (üblicherweise per Proc-Far-Definition), zum anderen muß das Programm die Parameter selbst vom Stack entfernen (in diesem Fall also mit »ret 4«).

(Wolfgang Lorenz/zi)

Name	datgen.pas
Sprache	Turbo Pascal ab Version 4.0
Grafikkarte	jede
Besonderheiten	benötigt Unit »dechex.pas«

```
[Z35]    1: { Programmname: DATGEN.PAS   }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {           DOS International }
[ØØØ]    4:
[HL3]    5: {$R-,S-,I-,D-,F-,V-,B-}
[ØØØ]    6:
[ASØ]    7: Uses
[N22]    8:   Crt, DecHex;
[ØØØ]    9:
[6XØ]   1Ø: Const
[UW2]   11:   Sprache: Byte = 1;
[PK3]   12:   Wort: Boolean = False;
[324]   13:   Dezimal: Boolean = False;
[224]   14:   Anzahl: Boolean = False;
[244]   15:   Nummern: Boolean = False;
[8G4]   16:   Overwrite: Boolean = False;
[FG2]   17:   Daten: Word = 16;
[US2]   18:   Laenge: Word = 79;
[5N2]   19:   Zeile: Word = 1ØØ;
[VY2]   2Ø:   Schritt: Word = 1;
[ØØØ]   21:
[RNØ]   22: Var
[1L3]   23:   Eingabe: File Of Byte;
[BD2]   24:   Ausgabe: Text;
[RK6]   25:   Param, EinName, AusName, Aus, S: String;
[RY2]   26:   X, Y, Fehler: Word;
[8V1]   27:   I: Integer;
[WY1]   28:   Taste: Char;
[F83]   29:   DateiEnde: Boolean;
[ØØØ]   3Ø:
[ML6]   31: Procedure HoleZahl (Var Z: Word; Unten,
[MW1]   32:                     Oben: Word);
[MUØ]   33: Begin
[1Y2]   34:   Delete (Param, 1, 1);
[AL3]   35:   Val (Param, X, Fehler);
[HV2]   36:   If Fehler <> 1 Then
[MUØ]   37:   Begin
[9R2]   38:     If Fehler > Ø Then
[LS4]   39:       Val (Copy (Param, 1, Fehler - 1),
[WP1]   4Ø:            X, Fehler);
[HR4]   41:     If (X < Unten) Or (X > Oben) Then
[MUØ]   42:     Begin
[T96]   43:       Writeln (^J'Numerischer Parameter '+
[4V4]   44:                'außerhalb des zulässigen '+
[LM3]   45:                'Bereichs von ', Unten,
[A22]   46:                ' bis ', Oben);
[W31]   47:       Halt (1)
[ZNØ]   48:     End;
[D31]   49:     Z := X;
[794]   5Ø:     While (Length (Param) > Ø) And
[9V3]   51:           (Param [1] in ['Ø'..'9']) Do
[BU2]   52:       Delete (Param, 1, 1)
[4MØ]   53:   End
[ZNØ]   54: End;
[ØØØ]   55:
[ØN2]   56: Procedure Ende;
[MUØ]   57: Begin
[PZ5]   58:   Writeln ('« trat ein Fehler auf');
[LE2]   59:   Close (Eingabe);
[PF2]   6Ø:   Close (Ausgabe);
[ME2]   61:   Erase (Ausgabe);
[W31]   62:   Halt (1)
[ZNØ]   63: End;
[ØØØ]   64:
[LE3]   65: Function Wert: Byte;
[MZ1]   66:  Var X: Byte;
[MUØ]   67: Begin
[773]   68:   If Eof (Eingabe) Then
[5SØ]   69:     X := Ø
[WP1]   7Ø:   Else Begin
[GL2]   71:     Read (Eingabe, X);
[5A3]   72:     If IOResult <> Ø Then
[MUØ]   73:     Begin
[F44]   74:       Write (^J'Beim Lesen aus '+
[CK3]   75:              'der Datei »', EinName);
[9PØ]   76:       Ende
[4MØ]   77:     End
[ZNØ]   78:   End;
[BE1]   79:   Wert := X
[ZNØ]   8Ø: End;
[ØØØ]   81:
[CK3]   82: Procedure LeseDatum;
[MUØ]   83: Begin
[TH1]   84:   X := Wert;
[9Z4]   85:   If Wort Then X := X + Wert Shl 8;
[N34]   86:   If Dezimal Then Str (X, S)
[WP1]   87:   Else Begin
[112]   88:     If Wort Then
[7S1]   89:       S := Hexw (X)
[ZRØ]   9Ø:     Else
[6V1]   91:       S := Hex (X);
[UM2]   92:     Case Sprache Of
[151]   93:       Ø: Begin
[MS3]   94:            If X >= Y Then S := 'Ø' + S;
[BG1]   95:            S := S + 'H'
[ZNØ]   96:          End;
[PZ1]   97:       1: S := '&H' + S;
[ZU1]   98:       2: S := '$' + S
[4MØ]   99:     End
[ZNØ]  1ØØ:   End;
[UM2]  1Ø1:   Case Sprache Of
[M72]  1Ø2:     Ø, 1: S := S + ',';
[RX1]  1Ø3:     2: S := S + '/?'
[4MØ]  1Ø4:   End
[ZNØ]  1Ø5: End;
[ØØØ]  1Ø6:
[KH3]  1Ø7: Procedure Vorspann;
[MUØ]  1Ø8: Begin
[NH2]  1Ø9:   If Nummern Then
[MUØ]  11Ø:   Begin
[8V2]  111:     Str (Zeile, Aus);
[T72]  112:     Aus := Aus + ' ';
[FV3]  113:     Zeile := Zeile + Schritt
[4MØ]  114:   End
[ZRØ]  115:   Else
[YD1]  116:     Aus := '';
[UM2]  117:   Case Sprache Of
[TB2]  118:     Ø: If Wort Then
[ST2]  119:          Aus := Aus + '    DW '
[ZRØ]  12Ø:        Else
[FØ3]  121:         Aus := Aus + '    DB ';
[313]  122:     1: Aus := Aus + 'DATA '
[4MØ]  123:   End
[ZNØ]  124: End;
[ØØØ]  125:
[3Y4]  126: Function NeueZeile: Boolean;
[MUØ]  127: Begin
[Z82]  128:   If Anzahl Then
[PD3]  129:     NeueZeile := I = Daten
[ZRØ]  13Ø:   Else
[EB4]  131:     NeueZeile := Length (Aus) +
[ZA3]  132:                  Length (S) - 1 > Laenge
[ZNØ]  133: End;
[ØØØ]  134:
[5C3]  135: Procedure Ausgeben;
[MUØ]  136: Begin
[MS1]  137:   If I > Ø Then
[MUØ]  138:   Begin
[E52]  139:     Dec (Aus [Ø]);
[SU3]  14Ø:     Writeln (Ausgabe, Aus);
[5A3]  141:     If IOResult <> Ø Then
[MUØ]  142:     Begin
[DM4]  143:       Write (^J'Beim Schreiben in '+
[KL3]  144:              'die Datei »', AusName);
[9PØ]  145:       Ende
[ZNØ]  146:     End;
[7RØ]  147:     I := Ø
[4MØ]  148:   End
[ZNØ]  149: End;
[ØØØ]  15Ø:
[FY3]  151: Procedure Hinzufuegen;
[MUØ]  152: Begin
[YV4]  153:   If Aus [Length (Aus)] = '?' Then
[E52]  154:     Dec (Aus [Ø]);
[Ø32]  155:   Aus := Aus + S
[ZNØ]  156: End;
[ØØØ]  157:
[MUØ]  158: Begin
[9B3]  159:   If (ParamCount = Ø) Or
[FR4]  16Ø:      (Pos ('?', ParamStr (1)) > Ø) Then
[MUØ]  161:   Begin
[BF6]  162:     Writeln (^J'DATGEN Eingabe [Ausgabe] '+
```

Listing 1. Das Quellprogramm »datgen.pas« verknüpft Obj-Dateien mit einer Hochsprache

```
[AG2]  163:              '[Optionen]');
[C54]  164:      Writeln ('  /A Assembler');
[235]  165:      Writeln ('  /B Basic (Standard)');
[TG3]  166:      Writeln ('  /P Pascal');
[KW5]  167:      Writeln ('  /O Keine Rückfrage bei '+
[993]  168:               'bestehender Datei');
[DH5]  169:      Writeln ('  /D Daten in dezimaler '+
[W41]  170:               'Form');
[T25]  171:      Writeln ('  /W Daten als Worte');
[MN5]  172:      Writeln ('  /N[#] Anzahl der Daten '+
[1L2]  173:               'pro Zeile (16)');
[2N5]  174:      Writeln ('  /L[#] Maximale Länge '+
[3F4]  175:               'einer Zeile (79, Standard)');
[5V4]  176:      Writeln ('  /Z[#[,#]] Zeilen '+
[594]  177:               'durchnummerieren (100, 1)');
[W31]  178:      Halt (1)
[ZNØ]  179:    End;
[SS3]  180:    EinName := ParamStr (1);
[EV4]  181:    For I := 1 To Length (EinName) Do
[XR5]  182:      EinName [I] := Upcase (EinName [I]);
[RM1]  183:    Param := '';
[GB4]  184:    For I := 2 To ParamCount Do
[4H4]  185:      Param := Param + ParamStr (I);
[ØV4]  186:    For I := 1 To Length (Param) Do
[D55]  187:      Param [I] := Upcase (Param [I]);
[942]  188:    AusName := '';
[794]  189:    While (Length (Param) > Ø) And
[1R2]  190:          (Param [1] <> '/') Do
[MUØ]  191:    Begin
[3Ø5]  192:      AusName := AusName + Param [1];
[BU2]  193:      Delete (Param, 1, 1)
[ZNØ]  194:    End;
[234]  195:    While (Length (Param) > 1) Do
[MUØ]  196:    Begin
[1Y2]  197:      Delete (Param, 1, 1);
[SL2]  198:      Case Param [1] Of
[NL2]  199:        'A': Sprache := Ø;
[UL2]  200:        'B': Sprache := 1;
[6N2]  201:        'P': Sprache := 2;
[7T3]  202:        'O': Overwrite := True;
[273]  203:        'D': Dezimal := True;
[XT2]  204:        'W': Wort := True;
[BF1]  205:        'N': Begin
[4H2]  206:               Anzahl := True;
[WK3]  207:               HoleZahl (Daten, 1, 255)
[ZNØ]  208:             End;
[5F1]  209:        'L': Begin
[PK2]  210:               Anzahl := False;
[TS3]  211:               HoleZahl (Laenge, 1Ø, 255)
[ZNØ]  212:             End;
[FG1]  213:        'Z': Begin
[5R2]  214:               Nummern := True;
[L34]  215:               HoleZahl (Zeile, Ø, 65535);
[9Z3]  216:               If (Length (Param) > Ø) And
[U23]  217:                  (Param [1] = ',') Then
[W54]  218:                 HoleZahl (Schritt, 1, 65535)
[4MØ]  219:             End
[ZNØ]  220:      End;
[794]  221:      While (Length (Param) > Ø) And
[1R2]  222:            (Param [1] <> '/') Do
[BU2]  223:        Delete (Param, 1, 1)
[ZNØ]  224:    End;
[S74]  225:    Assign (Eingabe, EinName);
[WF2]  226:    Reset (Eingabe);
[5A3]  227:    If IOResult <> Ø Then
[MUØ]  228:    Begin
[6N5]  229:      Writeln (^J'Die Eingabedatei »',
[U84]  230:               EinName, '« kann nicht '+
[V63]  231:               'geöffnet werden');
[W31]  232:      Halt (1)
[ZNØ]  233:    End;
[A64]  234:    If Length (AusName) = Ø Then
[MUØ]  235:    Begin
[C93]  236:      I := Pos ('.', EinName);
[743]  237:      If I = Ø Then I := 255;
[885]  238:      AusName := Copy (EinName, 1, I - 1)
[ZNØ]  239:    End;
[794]  240:    If Pos ('.', AusName) = Ø Then
[UM2]  241:      Case Sprache Of
[2G4]  242:        Ø: AusName := AusName + '.ASM';
[2G4]  243:        1: AusName := AusName + '.BAS';
[GA4]  244:        2: AusName := AusName + '.PAS'
[ZNØ]  245:      End;
[HA4]  246:    Assign (Ausgabe, AusName);
[6G3]  247:    If Not Overwrite Then
[MUØ]  248:    Begin
[ZG2]  249:      Reset (Ausgabe);
[263]  250:      If IOResult = Ø Then
[MUØ]  251:      Begin
[PF2]  252:        Close (Ausgabe);
[RY5]  253:        Write (^J'Soll die Datei »', AusName,
[ES4]  254:               '« überschrieben werden? ');
[P31]  255:        Repeat
[E24]  256:          Taste := Upcase (ReadKey)
[P24]  257:        Until Taste In ['J', 'N'];
[2V2]  258:        If Taste = 'N' Then
[MUØ]  259:        Begin
[VP2]  260:          Writeln ('Nein');
[LE2]  261:          Close (Eingabe);
[RPØ]  262:          Halt
[ZNØ]  263:        End;
[T62]  264:        Writeln ('Ja')
[4MØ]  265:      End
[ZNØ]  266:    End;
[Y13]  267:    Rewrite (Ausgabe);
[5A3]  268:    If IOResult <> Ø Then
[MUØ]  269:    Begin
[LE2]  270:      Close (Eingabe);
[PP5]  271:      Writeln (^J'Die Ausgabedatei »',
[AA4]  272:               AusName, '« kann nicht '+
[V63]  273:               'geöffnet werden');
[W31]  274:      Halt (1)
[ZNØ]  275:    End;
[112]  276:    If Wort Then
[781]  277:      Y := $AØØØ
[ZRØ]  278:    Else
[D41]  279:      Y := $AØ;
[LYØ]  280:    I := Ø;
[LT1]  281:    LeseDatum;
[P31]  282:    Repeat
[U84]  283:      DateiEnde := Eof (Eingabe);
[8N3]  284:      If I = Ø Then Vorspann;
[272]  285:      Hinzufuegen;
[K51]  286:      Inc (I);
[LT1]  287:      LeseDatum;
[Y94]  288:      If NeueZeile Then Ausgeben
[DS2]  289:    Until DateiEnde;
[WH1]  290:    Ausgeben;
[LE2]  291:    Close (Eingabe);
[VD2]  292:    Close (Ausgabe)
[KNØ]  293: End.
```

Listing 1. Das Quellprogramm »datgen.pas« verknüpft Obj-Dateien mit einer Hochsprache

```
[635]    1: { Programmname: DECHEX.PAS    }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[932]    5: Unit DecHex;
[ØØØ]    6:
[SK1]    7: Interface
[ØØØ]    8:
[DB5]    9:   Function Hex (Dec: Byte): String;
[DH5]   10:   Function Hexw (Dec: Word): String;
[ØØØ]   11:
[4L2]   12: Implementation
[ØØØ]   13:
[DB5]   14:   Function Hex (Dec: Byte): String;
[MUØ]   15:   Begin
[881]   16:     Inline (
[WW4]   17: $C4/$7E/$Ø8/    { LES   DI,[BP+Ø8] }
[354]   18: $BØ/$Ø2/        { MOV   AL,Ø2      }
[H24]   19: $AA/            { STOSB            }
[HW4]   20: $8A/$46/$Ø6/    { MOV   AL,[BP+Ø6] }
[L54]   21: $B1/$Ø4/        { MOV   CL,Ø4      }
[ØB4]   22: $D2/$E8/        { SHR   AL,CL      }
[G24]   23: $Ø4/$9Ø/        { ADD   AL,9Ø      }
[9N3]   24: $27/            { DAA              }
[F14]   25: $14/$4Ø/        { ADC   AL,4Ø      }
[9N3]   26: $27/            { DAA              }
[H24]   27: $AA/            { STOSB            }
[HW4]   28: $8A/$46/$Ø6/    { MOV   AL,[BP+Ø6] }
[D54]   29: $24/$ØF/        { AND   AL,ØF      }
[G24]   30: $Ø4/$9Ø/        { ADD   AL,9Ø      }
[9N3]   31: $27/            { DAA              }
[F14]   32: $14/$4Ø/        { ADC   AL,4Ø      }
[9N3]   33: $27/            { DAA              }
[B24]   34: $AA)            { STOSB            }
[ZNØ]   35:   End;
[ØØØ]   36:
[DH5]   37:   Function Hexw (Dec: Word): String;
[MUØ]   38:   Begin
[881]   39:     Inline (
[WW4]   40: $C4/$7E/$Ø8/    { LES   DI,[BP+Ø8] }
[D54]   41: $BØ/$Ø4/        { MOV   AL,Ø4      }
[H24]   42: $AA/            { STOSB            }
[DX4]   43: $8B/$46/$Ø6/    { MOV   AX,[BP+Ø6] }
[YE4]   44: $B9/$Ø4/$Ø4/    { MOV   CX,Ø4Ø4     }
[YG4]   45: $D3/$CØ/     { Ø: ROL   AX,CL      }
[XZ3]   46: $5Ø/            { PUSH  AX         }
[D54]   47: $24/$ØF/        { AND   AL,ØF      }
[G24]   48: $Ø4/$9Ø/        { ADD   AL,9Ø      }
[9N3]   49: $27/            { DAA              }
[F14]   50: $14/$4Ø/        { ADC   AL,4Ø      }
[9N3]   51: $27/            { DAA              }
[H24]   52: $AA/            { STOSB            }
[KX3]   53: $58/            { POP   AX         }
[K54]   54: $FE/$CD/        { DEC   CH         }
[R14]   55: $75/$EF)        { JNZ   Ø          }
[ZNØ]   56:   End;
[ØØØ]   57:
[KNØ]   58: End.
```

Listing 2. Die Hilfs-Unit »dechex.pas« dient der Zahlenkonvertierung

jetzt auch im Abo

DOS EXTRA 6: Alles über Basic

EXTRA Schneller Basic-Compiler zum Abtippen

EXTRA GW-Basic selbst erweitert

EXTRA Basic intern: Token, Variablen, Speicherverwaltung

EXTRA Datenbankprogramm mit Menüsteuerung

EXTRA Turbo-Basic-Funktions-Bibliothek für die CGA-Karte

EXTRA Strukturierte Programmierung in Quick Basic 4.0

...und vieles mehr!

Best.-Nr. 383 18,– DM*

Noch über den Verlag erhältlich:

DOS EXTRA 1: Ausverkauft

DOS EXTRA 2: Ausverkauft

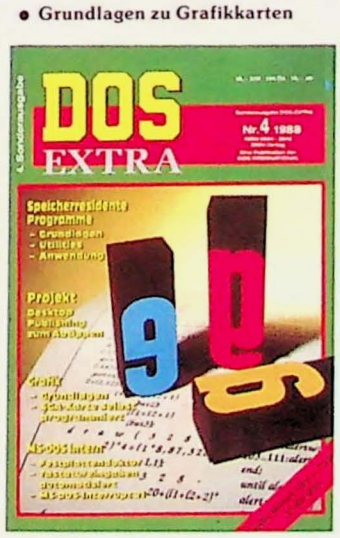
DOS EXTRA 3:
Für glasklare Informationen

- Der große DOS-Kurs »Der PC – glasklar in überarbeiteter Form
- Datenfernübertragungs-Programm in 100% Assemblercode
- Sprachdigitalisierung mit dem PC
- Hercules-Grafik-Toolbox in Turbo Pascal
- Grafikfunktionen für Turbo C

Best.-Nr. 356 18,– DM*

DOS EXTRA 4:
Was Sie schon immer interessiert hat

- Desktop Publishing selbstgemacht mit SDTP
- Speicherresidente Programme schreiben, analysieren und entfernen
- Alles über MS-DOS-Interrupt 21 hex
- Grundlagen zu Grafikkarten



Best.-Nr. 381 18,– DM*

DOS EXTRA 5:
Von Basic bis BIOS

- Aufbau und Programmierung der wichtigsten Grafikkarten
- Basic für Einsteiger und Profis
- Die Interrupts des BIOS im Detail
- Gerätetreiber-Grundlagen

Best.-Nr. 382 18,– DM*

Die Ausgaben DOS EXTRA 3, DOS EXTRA 4 und DOS EXTRA 5 sind nur noch über den Verlag zu beziehen. Zu allen Ausgaben ist jedoch die Databox im Format 5 1/4" oder 3 1/2" MS-DOS erhältlich. Bitte benutzen Sie die Bestellkarte.

* Unabhängig von der Anzahl der bestellten Zeitschriften berechnen wir für das Inland 4,– DM bzw. für das Ausland 6,– DM Porto und Verpackung.

DMV Verlag · Postfach 250 · 3440 Eschwege

Residenten Programmen auf der Spur

Speicherresidente Programme klinken sich in die Programm-Abläufe im Computer ein, indem sie Interrupt-Vektoren »verbiegen« und so beim Aufruf eines Interrupts selbst aktiv werden. Wollen Sie nun die Funktion eines solchen Programms nachvollziehen, so müssen Sie wissen, welche Vektoren wohin verbogen wurden. Schön wäre es, auch gleich die Bedeutung der umgeleiteten Interrupt-Vektoren zu erfahren. Und wenn Sie schließlich auch noch wissen möchten, wieviel Speicherplatz das residente Programm belegt, dann benötigen Sie das Programm »Chkints«.

Speicherresidente Programme sind eine nützliche Hilfe für den PC-Anwender, der mit ihnen leicht die Leistungsfähigkeit und Bedienungsfreundlichkeit seines Rechners erhöhen kann. Diese Programme werden entweder auf einen bestimmten Tastendruck hin vom Benutzer gestartet, beispielsweise Sidekick und Homebase, oder aber sie bilden Erweiterungen des Betriebssystems und fristen ihr Dasein überwiegend im Dunkeln wie der MS-DOS-Drucker-Spooler »Print«. In jedem Fall aber sind sie nur unter bestimmten Umständen aktiv. Man nennt sie deshalb auch Hintergrundprogramme.

So schön diese Hintergrundprogramme auch sind, so haben sie doch zwei entscheidende Nachteile: Sie zehren nämlich einerseits vom sowieso schon oft zu kleinen Arbeitsspeicher und kommen sich andererseits des öfteren gegenseitig in die Quere. So geschieht es, daß eine bestimmte Kombination von installierten Programmen den Rechner zum Absturz bringt.

Falls man sich nun einen Reim darauf machen möchte, warum ausgerechnet die Kombination Programm X1 und Programm Y2 den Rechner »ins Nirwana schickt«, so muß man zuerst einmal wissen, welche Interrupt-Vektoren beide Programme benutzen. Daraus lassen sich dann Rückschlüsse ziehen.

Aber auch, wenn Sie einfach nur ein bißchen neugierig sind und wissen wollen, wie ein bestimmtes Hintergrundprogramm funktioniert, oder es gar disassemblieren wollen, benötigen Sie Kenntnisse darüber, wie und wann es aufgerufen wird. Schon ist man wieder bei der Suche nach den verborgenen Interrupt-Vektoren angelangt.

```
F:\>chkints /u

CHKINTS vers 1.3      (C) 1988  DOS International

Neue Datei INTS.ADR erfolgreich erstellt!

F:\>graphics

F:\>chkints /c

CHKINTS vers 1.3      (C) 1988  DOS International

Unterschiede zwischen alter und neuer Interrupt-Vektor-Tabelle:
 Altes Code-Segment: 1D75h, aktuelles Code-Segment: 1E46h
    Unterschied in Bytes: 3344
 Interrupt: 00h:  Alter Vektor: 027F:5705, neuer Vektor: 1D85:1389
    HW: Division durch Null
 Interrupt: 05h:  Alter Vektor: 1D32:012A, neuer Vektor: 1D85:011C
    HW: Print Screen / ab 80186: BOUND

F:\>
```

Bild 1. Chkints zeigt, welche Interrupt-Vektoren von einem Hintergrundprogramm umgeleitet werden und wieviel Platz dieses belegt

Um nun herauszufinden, welche Interrupts von einem Programm umgeleitet werden, geht man in drei Schritten vor:

- Zuerst merkt man sich die Tabelle der Interrupt-Vektoren in ihrem Zustand vor dem Aufruf des residenten Programms,
- dann installiert man das zu untersuchende Programm resident im Speicher,
- und schließlich untersucht man die bei der Installation geänderte Tabelle von Interrupt-Vektoren auf Unterschiede zu der alten, die man sich »gemerkt« hat.

Stehen keine sonstigen Hilfsmittel zur Verfügung, so kann man zur Untersuchung der Vektoren das Programm »Debug« benutzen. Die Vektoren findet man im Speicher ab der Adresse 0000:0000hex (jeder Vektor belegt vier Byte). Zum »Speichern« der Information können Sie Papier benutzen. Ein bißchen umständlich, meinen Sie? Genau. Deshalb wurde für diese Arbeit ein Programm geschrieben, das all dies (und noch etwas mehr) selbständig durchführt.

Bild 1 zeigt die Anwendung des Programms. Wie beschrieben geht man dabei in drei Schritten vor: Mit dem Aufruf »chkints /w« wird die aktuelle Belegung der Interrupt-Vektor-Tabelle unter dem Namen »ints.adr« (keine ASCII-Datei) im aktuellen Unterverzeichnis (Directory) gespeichert. Dann ist das zu untersuchende Programm, in unserem Fall die Hardcopy-Routine »graphics« von MS-DOS, zu installieren. Zum Schluß wird noch einmal »chkints« aufgerufen, diesmal jedoch mit dem Parameter »/c«. Das Programm liest jetzt die alte Tabelle ein und vergleicht sie mit der aktuellen. Zeigen sich Unterschiede, so werden der zugehörige Interrupt sowie seine alte und neue Adresse auf dem Bildschirm ausgegeben.

Kein umständlicher Zahlensalat – die Ausgabe der Interrupt-Funktionen geschieht im Klartext

Als Besonderheit erläutert das Programm Chkint die Funktion des Interrupt im Klartext. So kann man beispielsweise Bild 1 entnehmen, daß »graphics« die Vektoren der Interrupts 00hex und 05hex umleitet. Der erste Interrupt behandelt den Fehler »Division durch Null«, der zweite wird aufgerufen, wenn der Benutzer die Taste [PrtSc] beziehungsweise [Druck] betätigt.

Daraus kann man schließen, daß die Hardcopy-Routine durch Drücken der PrtSc-Taste gestartet wird, was allerdings niemanden wundern wird. Interessanter ist dagegen schon, daß die Routine die eigenen Division-durch-Null-Fehler abfängt und dann selbst behandelt.

Dazu noch ein weiteres Beispiel, das MS-DOS-Programm »Print«. Dabei handelt es sich um einen Drucker-Spooler. Dieser Spooler lädt im Hintergrund, also praktisch nebenbei, eine ganze Liste von Dateien – eine nach der anderen – von der Diskette oder der Festplatte und überträgt sie Byte für Byte an einen angeschlossenen Drucker. All das wird erledigt, während der Anwender im Vordergrund weitgehend ungestört sein Pascal-Programm editiert oder schon den nächsten Brief eintippt.

Wichtig ist, daß kein anderes Programm auf den Drucker zugreifen darf, während »Print« aktiv ist. Dafür sorgt der Spooler, indem er die beiden Interrupts 05hex (Print Screen) und 17hex (Drucker-Ausgabe) umleitet und so unerwünschte Aufrufe abfängt. Das gleiche gilt für den BIOS-Interrupt 14hex (serielle Schnittstelle), denn »Print« kann einen Drucker auch über die serielle Schnittstelle bedienen, und auch dort darf kein anderes Programm dazwischenfunken.

Nun muß »Print« aber auch regelmäßig auf die Diskette beziehungsweise Festplatte zugreifen, um weitere Daten in den internen Puffer zu lesen. Damit auch hier andere

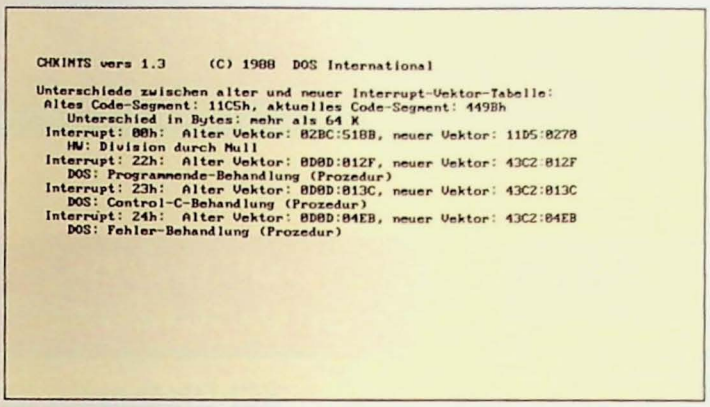


Bild 2. Diese Interrupts leitet Turbo Pascal 4.0 um, wenn man ein Programm aus der »OS-Shell« startet

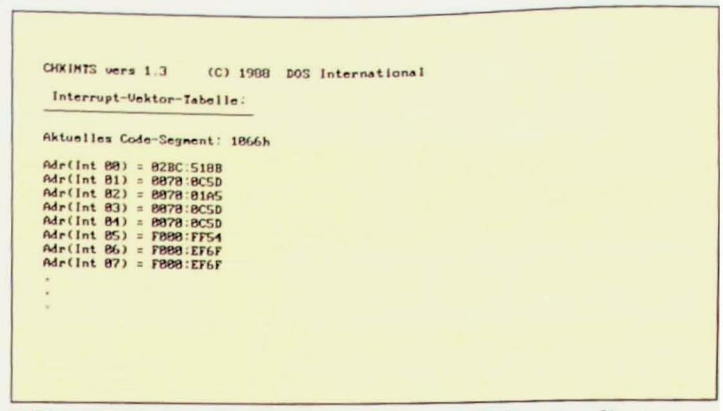


Bild 3. Der Beginn der Liste aller Interrupt-Vektoren, die man mit dem Kommando »chkints /l« erhält

Programme nicht gleichzeitig einen Zugriff versuchen und so ein heilloses Durcheinander anrichten, wird vom Spooler der Interrupt 13hex (Diskette und Festplatte) umgeleitet. Auf diese Weise erkennt »Print« einerseits, ob gerade ein Diskettenzugriff aktiv ist und stellt seine eigene Anforderung zurück. Andererseits lassen sich aber auch fremde Programme zum Warten zwingen, solange das Hintergrundprogramm selbst auf eine Datei zugreift.

Das erneute Booten von Diskette beispielsweise (Interrupt 19hex – »Warm Boot«) würde den Spooler empfindlich in seiner Funktion stören – der Interrupt wird deshalb abgefangen und seine Ausführung entsprechend verzögert.

Etwas schwieriger zu durchschauen sind die Verhältnisse beim Interrupt 15hex, der eine ganze Reihe verschiedener Unterfunktionen besitzt, die sich eventuell noch bei verschiedenen Computertypen unterscheiden. Wahrscheinlich wird mit dem Verbiegen dieses Interrupts die Bedienung der Sysreq-Taste oder das Umschalten in den »protected mode« bei AT-kompatiblen Rechnern verzögert.

Leicht zu erklären ist dagegen die Bedeutung des Interrupt 1Chex (Timer-Interrupt) für den Spooler. Diese Zeitgeber-Funktion wird genutzt, um den Spooler periodisch immer wieder zu aktivieren. Er überprüft dann den Status des angeschlossenen Druckers und übermittelt ihm gegebenenfalls ein neues Zeichen.

Die Beispiele zeigen, welchen Nutzen das Programm Chkints bringt und wie man die zur Verfügung gestellten Informationen interpretiert. In Bild 2 finden Sie noch ein letztes Beispiel: Diesmal mußte die Version 4.0 von Turbo Pascal für ein Experiment herhalten. Es wird dabei zuerst mit »chkints /w« die Interrupt-Vektor-Tabelle gespeichert und dann Turbo Pascal aufgerufen. Von Turbo Pascal aus wird dann Chkints als Kind-Prozeß aus der OS-Shell erneut gestartet – diesmal mit der Option »/c«. Das Ergebnis sieht man in Bild 2. Verläßt man Turbo Pascal wieder, so haben alle Interrupt-Vektoren die alten Werte.

Ein paar Sonderfälle gibt es noch bei den Interrupt-Vektoren; das heißt, genau genommen handelt es sich hierbei um keine wirklichen Interrupt-Vektoren, sondern vielmehr um Zeiger auf Datenbereiche (Interrupt 1Dhex bis Interrupt 1Fhex) beziehungsweise auf Prozeduren des Far-Typs (Interrupt 22hex bis Interrupt 24hex).

Chkints macht nur geringe Platzansprüche für den Eigenbedarf geltend und gibt ungenutzten Speicher frei

Das Programm Chkints finden Sie in Listing 1. Dem Programmkopf ist zu entnehmen, mit welchen Befehlen die Quell-Datei zu übersetzen ist. Die Fehlermeldung »No Stack Segment« des Linkers dürfen Sie wie üblich ignorieren. Dem Programmkopf entnehmen Sie weiterhin, mit welchen Parametern das fertige Programm versorgt werden muß. Sollten Sie dies einmal vergessen haben, so bewirkt der Aufruf von Chkints ohne Parameter, daß die korrekte Syntax des Programms ausgegeben wird.

An dieser Stelle sei auch die Option »/l« von Chkints erwähnt. Wird sie beim Aufruf des Programms angegeben, so wird die gesamte Tabelle der Vektoren ausgegeben, allerdings ohne weitere Erläuterungen (Bild 3).

Das Programm gibt als erstes, nachdem es unter MS-DOS geladen und zur Ausführung gebracht wurde, den nicht benötigten Speicher mittels der Funktion 4Ahex (Größe eines Speicherblocks festlegen) wieder frei. Denn unabhängig von der jeweiligen Größe reserviert MS-DOS für jedes Programm, das als Com-Datei geladen wird, ganze 64 KByte.

Chkints benutzt den Wert des Code-Segments, um den »Speicherhunger« eines Hintergrundprogramms zu berechnen. Dazu wird beim ersten Aufruf der Wert dieses Segments in der Datei »ints.adr« gespeichert und beim zweiten Aufruf mit dem aktuellen Wert verglichen. Zeigt sich ein Unterschied, so muß sich ein weiteres Programm zwischen der alten und der neuen Segmentadresse eingenistet haben. Die Größe des speicherresidenten Programms läßt sich deshalb leicht aus der Differenz der beiden Segmentwerte berechnen. Man darf dabei allerdings nicht vergessen, daß es auch Programme gibt, die sich selbst am oberen Ende des RAM-Bereichs installieren. Deren Größe kann man auf diese Weise natürlich nicht in Erfahrung bringen. Mit Programmen dieser Art werden Sie aber nur selten konfrontiert.

Die berechnete, von residenten Programmen benutzte Speichergröße wird dezimal ausgegeben, falls der Unterschied der Segmentwerte nicht größer als 64 KByte ist. Ist er jedoch größer, so wird nur eine entsprechende Meldung, nicht aber der Wert angezeigt. Das geschieht deshalb, weil die Umrechnung einer größeren Hexadezimal-Zahl in einen ASCII-String wesentlich mehr Programmieraufwand erfordert hätte.

Ein Tip am Rande: Die meisten Hintergrundprogramme lassen sich nur ein einziges Mal installieren, nämlich mit ihrem ersten Aufruf. Werden sie ein zweites Mal aufgerufen, so mögen sie zwar dabei auch irgendeine Funktion erfüllen, neu im Speicher installiert werden sie normalerweise jedoch nicht. So würde beispielsweise die Befehlsfolge

```
graphics
chkints /w
graphics
chkints /c
```

keine umgeleiteten Interrupts zu Tage fördern. Diese Tatsache gilt es besonders bei der Untersuchung von Programmen zu beachten, die bereits aus der Datei »autoexec.bat« aufgerufen werden. Für eine Überprüfung sind sie von dort dann erst zu entfernen.

DMV-Versandservice

Die preiswerte Alternative für aktive PC-Anwender!

Textverarbeitung

ConText - Textverarbeitung mit Konzept

Kompromißlose WYSIWYG-Darstellung im Grafikmodus:
Fett-, Kursiv- und Breitschrift "life" auf dem Bildschirm ohne Störung durch Steuerzeichen

Einfache und blitzschnelle Bedienung:
Pulldown-Menüs und Shortcut- Bedienung, Hilfs-Fenster und Mausunterstützung in jeder Arbeitsphase

Vielseitigkeit und hohes Tempo:
alle Textverarbeitungs-Standardfunktionen plus zahlreiche unkonventionelle, durchdachte Extras

Flexibilität:
frei konfigurierbare Tastenbelegung, Bildschirmdarstellung, Layoutgrundeinstellung, Druckeransteuerung und eigene Zeichensätze

ConText Pro

Das Textprogramm für den professionellen Einsatz
(M)(D) DM **199,–**

ConText PC

Erprobte Version für den Einstieg
(D) DM **99,–**

Upgrade auf ConText Pro

bei Einsendung des Handbuches ConText PC
DM **129,–**

Zusatzprogramme:

ConText-ADREWA

Komfortabel speicherresident, auch für beliebige andere Programme nutzbar
(L) DM **49,–**

ConFont

Vektor-Zeichensatzeditor mit Druckerdownload für ConText PRO
(L) DM **99,–**

DMV-Font I

Times Roman, Triplex, griechisch-mathematischer Sonderzeichensatz
(M)(L) DM **99,–**

InfoWare

Ein Produkt, das sich bereits einen Namen gemacht hat: InfoWare
Speicherresidentes Online-Informationssystem mit Querverweisen und eigenen Einträgen für folgende Produkte und Themen: Lotus 1-2-3, Quick Basic, Turbo Basic, GW Basic, Quick C, Turbo C, Assembler 8086/88, MS-Word 3.0/4.0 Grundlagen, WordPerfect 4.2 - Grundlagen,
PC-Grundlagen, C-Grundlagen, MS-DOS, PC-DOS,
(D) DM **69,–**
Modula II, Framework II, Fortran 77, dBase III Plus
DM **99,–**
System-Mindestvoraussetzung: 512 kByte RAM

Nicht mehr lieferbar

Anwendungen

ERGO

DOS-Benutzeroberfläche

Die Benutzeroberfläche für Profis mit minimal 12 kByte Speicherplatzbedarf. Inklusive: Fullscreen-Editor, Informationssystem, Arbeitshilfen (Taschenrechner, Kalender, etc.), Passwort-Hierarchie und integriertem Byte-Editor.
(M) (D) DM **149,–**

Hyperkey Plus 4.0

PC/XT/AT-Tastaturbelegung total. Speicherresidenter Macro-Prozessor mit Chaining, EMS-Unterstützung und speicherplatzsparender, Abspiel-Version.
(M) DM **99,–**
Update von Version 3.0 DM **29,–**

FontEdit II

Die Möglichkeiten von Vektorgrafik für eigene Zeichensätze nutzen: Bezier-Kurven-Technik, Vergrößern,
Verkleinern, Drehen und Spiegeln, Zusammenfügen und Kopieren.
Praktische Anwendung durch Treiber und Druckerdownload für Turbo Pascal 4/5, Turbo C ab 1.5, Microsoft C 5.0 und TopSpeed Modula-2.
Umsetzung in Pixelformate, z.B. für Epson 9-Nadel- und 24- Nadel-Drucker und kompatible, NEC 24-Nadel-Drucker, alle HP-Laserjet, Ventura Publisher, GEM/3, MS-Windows ab 2.0, etc.
(M)(L) DM **149,–**

Update von FontEdit II DM 59,–

DMV-Font I

Vektorfonts Times Roman, Triplex, griechisch-mathematischer Sonderzeichensatz
(L) DM **99,–**

VirusDoktor

Abwehr und Schutz vor Computerviren mit einer in der Praxis erprobten Programmsammlung: Fünf einzelne Programme leisten mehr als jede Komplettlösung. Das Handbuch informiert Sie in einem ausführlichen Grundlagenteil über Systematik, Auftreten und Bekämpfung von PC-Computerviren.
DM **99,–**

MaskEdit Plus

Maskenerstellung mit Maussteuerung, per Pulldown-Menü oder Funktionstasten: typgeprüfte Eingabe- und Ausgabefelder, Rahmen mit Eckenautomatik, Farbe durch Farbtabellen, kompletter ASCII-Zeichensatz. Teile einer Maske können kopiert, gelöscht, zentriert, verschoben und gefüllt, einzelne Zeilen eingefügt oder gelöscht werden. Voller Komfort durch Undo-Funktion und kontextbezogene Hilfe.

MaskEdit Plus I

für Turbo Pascal 3/4/5 und dBase III +
(M)(D) DM **99,–**

MaskEdit Plus II

für Turbo C und Microsoft C
(M)(D) DM **99,–**
Paketpreis I + II, (M)(D) DM **149,–**

Software-Entwicklung

BCI DisAsm

Disassemblieren, Analysieren und Dokumentieren von Maschinenprogrammen für alle 8088/8086/80186/80286/80386-CPUs ab DOS 2.0 in Microsoft-MASM-kompatiblem Assemblerlisting.
Intelligenter MS-DOS-Disassembler
DM **199,–**

TopSpeed Modula-2

Die neue Dimension vom legendären Entwickler von Turbo Pascal: Das schnellste Modula-System, das es je gab. Demnächst auch für OS/2 in kompatibler Version!
deutsche Version mit deutschem Handbuch
TopSpeed Compiler inklusive
ISAM & Display Manager DM **348,–**
TopSpeed TechKit DM **248,–**
TopSpeed Debugger DM **248,–**
Paketpreis für Compiler und Debugger zusammen DM **555,–**
OS/2-Version DM **698,–**

DOS-Software-Tools

Jetzt neu aus der Software-Schmiede der DOS International: High-Tech-Software zu Low-Cost-Preisen

DOS-Tools Volume 1

RSM - Resident-Software-Manager

Menügesteuertes Laden, Löschen und Nachladen von residenten (TSR) Programmen.
(M) DM **69,–**

DOS-Tools Volume 2

PC-Utilities

Zeitlupe PC, Softreset, universeller Tastaturtreiber »T6« mit Quellcode.
DM **69,–**

DOS-Tools Volume 3

MS-DOS-Utilities

Fünf neue DOS-Befehle: CALL für verschachtelte Aufrufe von Batch-Dateien, CODE verschlüsselt Dateien mit einem Passwort, MAKEROOM löscht in allen Unterverzeichnissen, SETDIR und RESTDIR, TY für Hex-Dump auf Drucker und Bildschirm, SCopy und Filed-Editor.
DM **49,–**

Databox

Zu den Zeitschriften DOS Int., Window, PC Amstrad Int., PCpur und TOOLBOX gibt es jeweils den beliebten Databox-Service mit den Programmen jeder Ausgabe auf Datenträger. Beachten Sie bitte die entsprechende Werbung in jedem Heft.
Jede Diskette DM **24,–**
Window-Databox DM **27,–**

Tausende zufriedener Kunden sind seit Jahren der Beweis: DMV-Software entsteht in enger Zusammenarbeit mit den verschiedenen Fachredaktionen des Hauses. Wer könnte kompetenter über die Qualität urteilen?
Wenn Sie nähere Informationen über unsere Produkte benötigen, fordern Sie bitte unseren Katalog an. Zu manchen unserer Produkte sind auch Demo-Disketten erhältlich, beachten Sie bitte die entsprechenden Hinweise!

TOOLBOX-Spezial

Beliebte Programme aus der TOOLBOX zum Anwenden, Lernen, Probieren und Studieren. Jeweils mit Quellcode und Dokumentation auf der Diskette zum Superpreis:

Spezial I: Turbo Prolog
Spezial II: PasComp - Standard Pascal Compiler
Spezial III: Echtzeitverarbeitung
Spezial IV: CALC & Kurvendiskussion
Spezial V: STRUKTO Plus "Pascal"
Spezial VI: Spiel & Spaß
Spezial VII: Deutsche Silbentrennung
Spezial VIII: Spieltheorie
Spezial IX: HPGL-Plotter
Spezial X: CRC-Softwaresicherung (L)
Spezial XI: MAP, MARK & RELEASE (L)
Spezial XII: Turbo Pascal 4/5 Tools (L)

Alle TOOLBOX-Spezial jeweils nur (D) DM **35,–**

Software-Tools

CGX- Color Grafik eXtension

für Schneider PC 1512 und Turbo Pascal 3.0
16 Farben, Linien, Kreise , beliebige Schriften, etc.
DM **49,–**

TOOLBOX-Professional

H.A.I.-Tech - Hochsprachen-Assembler-Interface

Sammlung von Makros, um von Hochsprachen aus betriebssystem-oder hardwarenahe Funktionen zu nutzen.
Folgende Compiler und Sprachen werden unterstützt: Lahey Fortran F77L, ab V.2.1, Microsoft Fortran ab V.4.0, Ryan/McFarland Fortran ab V.2.1, Prospero Fortran ab V.1.2, Turbo Pascal ab V.4.0, Microsoft Pascal ab V.3.2, Oregon Pascal-2 ab V.2.1, Turbo C ab V.2.0,
Microsoft C ab V.4.0, TopSpeed Modula-2 ab V.1.0.
Als Anfangsbestand und zur Einarbeitung werden ausgearbeitete Module über die MS-DOS-Funktionen mitgeliefert.
DM **199,–**

Bücher

Exklusiv für alle Amstrad/Schneider-Anwender
Die Basic 2 Toolbox DM **49,–**
Diskette zum Buch DM **29,–**

Aus dem Addison-Wesley-Angebot:
Arne Schäpers: Konzepte, Analysen, Tips & Tricks

Turbo Pascal 5.0	DM	**58,–**
Diskette zum Buch	DM	**91,20**
Turbo C	DM	**68,–**
Diskette zum Buch	DM	**91,20**
Das dBase-III-Handbuch	DM	**78,–**
PC-DOS - Eine Einführung	DM	**48,–**

Aus dem Sybex-Verlag:

Das QuickBASIC-Handbuch	DM	**49,–**
Das PC-Profibuch	DM	**49,–**

Weitere Bücher von Addison-Wesley und Sybex auf Anfrage

Unterhaltung

Nach der Arbeit das Vergnügen: Entspannung mit dem DMV-Unterhaltungsangebot!

PC-Spielebox No.1
BIMBO II, BOUNCER DM **49,–**

PC-Spielebox NO.2
JAKEY, STARVISION DM **49,–**

Underground-Aktion DM **49,–**

Unterhaltsames Frage- und Antwortspiel für die ganze Familie

Know PC DM **49,–**
Know-PC-Ergänzungsfragen DM **29,–**

Faszinierender Computer: Die Entdeckungsreise in das unerschöpfliche Wunderland von Fraktalgrafiken

Fraktal Generator 2-D DM **49,–**
Fraktal Generator 3-D DM **69,–**

Systemvoraussetzungen: EGA-Karte oder Amstrad/Schneider PC 1512, (D)

Sonderhefte

DOS Extra III bis VI jeweils	DM	**18,–**
DATABOX DOS Extra I	DM	**24,–**
DATABOX DOS Extra II	DM	**45,–**
DATABOX DOS Extra III bis VI	DM	**35,–**

Sonderdruck
PASCAL
Das Beste aus PASCAL 11'86 bis 6'87 DM **25,–**
Datenträger PASCAL Sonderdruck, Volume 1 und 2 DM **35,–**

Sammelmappen

Ordnung und Übersicht schaffen die beliebten DMV-Sammelmappen
für DOS, Window, PC Int., PCpur, JOYSTICK und TOOLBOX je 2 Stück DM **15,80**

Vertriebsprodukte

C-Funktions-Sammlung
für Turbo C und Microsoft C DM **99,–**

GEMInterface PC
Toolkit zur Erstellung eigener GEM-Applikationen, für

Turbo Pascal 4/5	DM	**198,–**
Logitech Modula-2	DM	**249,–**
TopSpeed Modula-2	DM	**198,–**
in Verbindung dazu GEM/3-Desktop	DM	**99,–**

Zeitschriften

Noch jahrelang aktuell sind die Zeitschriften aus dem Hause DMV! Vervollständigen Sie Ihre Sammlung! Einzelhefte laut Bestellkarte

Paketangebote

PC Amstrad Superpack 1/87 -12/87 + 3x86 + 2 Sammelordner	DM	**50,–**
Jeweils 6 PASCAL-Ausgaben freier Wahl, ab Heft 11'86 bis Heft 8/9'88	DM	**25,–**
Pack I: 7 Hefte, PASCAL 11'86 bis 6'87	DM	**28,–**
Pack II: 6 Hefte, PASCAL 7'87 bis 12'87	DM	**25,–**
Pack III: 7 Hefte, PASCAL 1'88 bis 8/9'88	DM	**28,–**

Amiga und Atari

High-Speed 3D-Fraktalgrafiken in Super-Parallel-Projektion mit phantastischen Farbmöglichkeiten.

Amiga Fraktal Generator 3-D	DM	**69,–**
Atari ST Fraktal Generator 3-D	DM	**69,–**

Bootblock-Viruskiller und Archivierung von Disketten-Tracks und -Blöcken für Commodore Amiga.

ViroTrack DM **59,–**
ab Kickstart Version 1.2

TOOLBOX Edition 68000
Sonderheft mit Listings für Fraktalgrafiken, Viruskiller, File-Requester, und andere. DM **18,–**

DATABOX für Atari ST und Amiga jeweils DM **29,–**

Unabhängig von der Anzahl der bestellten Produkte, Programme und Bücher berechnen wir für das Inland DM 4,– bzw. für das Ausland DM 6,– Porto und Verpackung.

Alle Produkte sind wahlweise auf 5 1/4-Zoll und 3 1/2-Zoll-Disketten erhältlich. Bitte beachten Sie die jeweiligen Bestellnummern. Bitte benutzen Sie die Bestellkarte und die darauf angegebenen Bestellnummern.
Soweit nicht anders vermerkt, sind alle Produkte auf MS-DOS-Computern ab DOS 2.1, minimal 256 kByte RAM und den Grafikkarten CGA, MGA, EGA, VGA und Hercules lauffähig.

Legende:
(M) = mit universell verwendbarem Maustreiber
(D) = Demo-Diskette für DM 5,– erhältlich
(L) = Liefertermin 1.6.1989

DMV-Verlag · Postfach 250 · 3440 Eschwege

DMV Daten- und Medienverlag

Das Programm Chkints hat nicht nur für neugierige »Systemanalytiker«, sondern auch für den »ganz normalen« PC-Anwender eine Bedeutung; und zwar zum Schutz vor Computer-Viren. Einige dieser unangenehmen »Lebewesen« machen sich nämlich auf die folgende Weise in Ihrem Computer breit:

Mit Chkints schützen Sie sowohl Ihren Computer als auch Ihre gesamten Programmbestände vor Infektionen

Ein argloser Computer-Benutzer startet ein Programm, daß er vielleicht gerade erst aus einer Mailbox geladen hat. Das Programm scheint wie gewohnt zu laufen und gibt keinerlei Grund zur Besorgnis. Doch gerade in dem Augenblick, in dem der Anwender die Arbeit mit diesem Programm beendet, wird der Virus – mit dem die Software infiziert ist – aktiv: Er sorgt für sein eigenes »Überleben«, indem er sich als residentes Programm im Speicher festsetzt, von dort aus andere Programme infiziert und dabei zusätzlich noch den einen oder anderen Schaden anrichtet.

Mit dem Programm Chkints können Sie leicht feststellen, ob sich ein Virus bei Ihrem Computer eingenistet hat: Rufen Sie zuerst Chkints mit dem Parameter »/w« auf. Nachdem die Interrupt-Vektor-Tabelle in der Datei »ints.adr« abgelegt wurde, starten Sie das verdächtige Programm. Nach dessen Ablauf geben Sie das Kommando »chkints /c« ein und sehen damit, ob ein unbekannter »Täter« irgend etwas im Speicher hinterlassen hat. Ist das der Fall, so sollten Sie schnellstens Ihren Rechner neu booten.

Am besten führen Sie dies »cold« aus, also per Reset-Taste oder, wenn diese fehlt, durch Aus- und erneutes Einschalten des Gerätes. Denn wer weiß, ob der Virus nicht gerade erst aktiv wird, wenn Sie die Tastenkombination [Ctrl-Alt-Del] ausführen?

(Frank Schmidt/zi)

Name	chkint.asm.
Sprache	MASM ab Version 4.0, TASM
Grafikkarte	jede
Besonderheiten	keine

```
[NK1]    1: page 60,132
[Ø82]    2: title chkints
[R39]    3: subttl Utility zum Untersuchen der Interrup
            t-Vektoren
[ØØØ]    4:
[2H1]    5: comment %
[TF6]    6: ========================================
[T73]    7:          (C) Everts & Hagedorn
[833]    8:              DOS International
[ØØØ]    9:
[CW4]   10: Programm:          CHKINTS vers 1.3
[2Ø3]   11: Datum:             24.Ø7.88
[T23]   12: Letzte Änderung:   31.Ø7.88
[YE4]   13: Autor:             Frank Schmidt
[NK3]   14: Assembler:         MASM 4.Ø
[ØØØ]   15:
[L46]   16: Funktion: Erstellt eine Liste der PC-
[XU5]   17:           Interrupt-Vektoren, speichert sie
[3H5]   18:           auf Diskette oder vergleicht die
[H34]   19:           Liste mit einer schon auf
[3G5]   2Ø:           Diskette abgespeicherten. Damit
[KR5]   21:           läßt sich leicht untersuchen,
[LK5]   22:           welche Interrupts von residenten
[CS4]   23:           Programmen verbogen werden.
[ØØØ]   24:
[6V2]   25: Syntax:   CHKINTS
[2F5]   26:             Hilfe: Korrekte Syntax ausgeben
[TS1]   27:           CHKINTS /L
[8Ø5]   28:             aktuelle Interrupt-Liste auf
[6C3]   29:             Bildschirm ausgeben
[UT1]   3Ø:           CHKINTS /W
[YX4]   31:             aktuelle Interrupt-Liste in
[FP2]   32:             Datei schreiben
[YR1]   33:           CHKINTS /C
[RZ3]   34:             aktuelle Liste mit Datei
[7X3]   35:             vergleichen und Unter-
[1D5]   36:             schiede auf Bildschirm ausgeben
[ØØØ]   37:
[A57]   38: Alle Parameter dürfen auch gemischt werden
[ØØØ]   39:
[7L6]   4Ø: Return Error Codes: 1: Falsche Kommando-
[XVØ]   41:                        Zeile
[KG3]   42:                     3: Pfad nicht gefunden
[8M3]   43:                        (beim Datei-Eröffnen
[C71]   44:                         mit /W)
[P53]   45:                     4: Zu viele Dateien
[1H1]   46:                        offen (   "   )
[CE3]   47:                     5: Zugriff verweigert
[ØØØ]   48:
[ØØØ]   49:
[9C6]   5Ø: Übersetzen mit: masm CHKINTS,,nul,nul
[Y53]   51:                 link CHKINTS,,nul,,
[KZ4]   52:                 exe2bin CHKINTS CHKINTS.COM
[KT2]   53:                 del CHKINTS.EXE
[ØØØ]   54:
[TF6]   55: ========================================
[A2Ø]   56: %
[ØØØ]   57:
[P42]   58: ; Interrupts
[3T1]   59:       msdos = 21h
[AK2]   6Ø:   GetIntVec = 35h
```

```
[197]   61: ; Funktionsnummer: Interrupt-Vektor lesen
[KF2]   62:   PrtStrg   = Ø9h
[YB7]   63: ; Zeichenkette auf Standard-Output ausgeben
[T82]   64:   PrtChr    = Ø2h
[N96]   65: ; Zeichen auf Standard-Output ausgeben
[XD2]   66:   CrHandle  = 3Ch
[KK4]   67: ; Datei erstellen/eröffnen
[KE2]   68:   OpHandle  = 3Dh
[422]   69: ; Datei eröffnen
[HD2]   7Ø:   ClHandle  = 3Eh
[3C3]   71: ; Datei schließen
[UD2]   72:   RdHandle  = 3Fh
[NP2]   73: ; Aus Datei lesen
[ED2]   74:   WrHandle  = 4Øh
[943]   75: ; In Datei schreiben
[2F2]   76:   SetBlock  = 4Ah
[2V6]   77: ; Größe eines Speicherblocks ändern
[3A2]   78:   RetDos    = 4Ch
[ØD5]   79: ; Rückkehr zum DOS (Programmende)
[ØØØ]   8Ø:
[H32]   81: ; Konstanten
[512]   82:   FirstInt  = Ø
[KM5]   83: ; erster Interrupt, der in Tabelle
[1Ø3]   84: ; aufgenommen wird
[ØG2]   85:   LastInt   = Ø7Fh
[7G3]   86: ; letzter Interrupt
[BG1]   87:   cr        = 13
[8V2]   88: ; Wagenrücklauf
[GF1]   89:   lf        = 1Ø
[3W2]   9Ø: ; Zeilenvorschub
[TF1]   91:   tab       = 9
[S12]   92: ; Tabulator
[GU1]   93:   etx       = '$'
[464]   94: ; Markierung String-Ende
[ØØØ]   95:
[4N1]   96: ; Adressen
[8B2]   97:   CmdSize   = Ø8Øh
[LF4]   98: ; Länge der Kommando-Zeile
[352]   99:   CmdBuf    = Ø81h
[6E5]  1ØØ: ; Beginn Kommando-Zeilen-Puffer
[ØØØ]  1Ø1:
[7Z5]  1Ø2: ; Segment-Definitionen für COM-File
[ØØØ]  1Ø3:
[633]  1Ø4:  cseg segment 'code'
[NN6]  1Ø5:  assume cs:cseg, ds:cseg, es:cseg, ss:cseg
[ØØØ]  1Ø6:
[CE1]  1Ø7:     org  Ø1ØØh
[ØØØ]  1Ø8:
[NF6]  1Ø9: ;========================================
[ØØØ]  11Ø:
[2H1]  111: comment %
[ØØØ]  112:
[LA7]  113: Kurze Übersicht über die Hauptprogramm-
[ZØ1]  114: Teile:
[ØØØ]  115:
[2M5]  116:  ChkInts:       Beginn des Programms &
[5M4]  117:                 Untersuchung der Kommando-
[XVØ]  118:                 Zeile
[WE6]  119:  BuildTab:      Interrupt-Vektor-Tabelle
[BF2]  12Ø:                 im RAM aufbauen
[FA6]  121:  ListTable:     Gesamte Vektor-Liste auf
```

```
[6C3]  122:                    Bildschirm ausgeben
[NB6]  123:  CompareTables:    Alte Vektor-Tabelle von
[1C3]  124:                    Platte einlesen, mit
[ØB4]  125:                    aktueller vergleichen und
[ØP3]  126:                    Unterschiede ausgeben
[536]  127:  WriteTable:       Neue Vektor-Tabelle auf
[BP2]  128:                    Platte schreiben
[A2Ø]  129: %
[NF6]  13Ø: ;=========================================
[ØØØ]  131:
[ET3]  132: ; Beginn Hauptprogramm
[ØØØ]  133:
[HF1]  134: ChkInts:
[NU1]  135:          jmp     start
[275]  136: ; Signon-Message überspringen
[ØØØ]  137:
[766]  138: Signon: db   cr,lf,'CHKINTS vers 1.3     '
[AY4]  139:         db   '(C) 1988  DOS International'
[TF2]  14Ø:         db   cr,lf,lf,etx
[ØØØ]  141:
[785]  142: start:  mov    sp,offset NewStack
[2Ø5]  143: ; Stackpointer initialisieren
[TU2]  144:         mov    ah,SetBlock
[5A6]  145: ; nicht benötigten Speicher freigeben
[ZH5]  146:         mov    bx,(NewStack-ChkInts)/16+11h
[CR5]  147: ; Länge des benutzten Paragraphen
[UU1]  148:         int    msdos
[D95]  149: ; ES zeigt auf benutztes Segment
[RP2]  15Ø:         mov    ah,PrtStrg
[F64]  151: ; Signon-Message ausgeben
[BV3]  152:         mov    dx,offset signon
[UU1]  153:         int    msdos
[ØØØ]  154:
[TV5]  155: Parse:  mov    byte ptr LstFlag,Øffh
[CY4]  156: ; Zuerst alle Flags löschen
[4G4]  157:         mov    byte ptr WrFlag,Øffh
[FL4]  158:         mov    byte ptr CmpFlag,Øffh
[9E2]  159:         mov    si,CmdBuf
[8K5]  16Ø: ; dann Kommando-Zeile untersuchen
[ØØØ]  161:
[CL1]  162: ParseLop:
[5D2]  163:         mov     al,[si]
[U53]  164: ; ein Zeichen lesen
[ZR1]  165:         cmp     al,' '
[HC2]  166: ; Leerzeichen?
[YS1]  167:         jz      NxChr
[F95]  168: ; dann gehe zu nächstem Zeichen
[ZX1]  169:         cmp     al,tab
[R32]  17Ø: ; Tabulator?
[YS1]  171:         jz      NxChr
[K93]  172: ; auch überspringen
[XS1]  173:         cmp     al,'/'
[K33]  174: ; Option-Schalter?
[6G2]  175:         jz      ChkOption
[DZ2]  176: ; dann untersuchen
[8N1]  177:         cmp     al,cr
[E24]  178: ; Ende der Zeile erreicht?
[Y62]  179:         je      ChkFlags
[914]  18Ø: ; dann weiter im Programm
[2H2]  181:         jmp     CmdLinErr
[5C5]  182: ; sonst Fehler in Kommando-Zeile,
[ØØØ]  183:
[WE1]  184: ;  Abbruch
[ØØØ]  185:
[RC2]  186: NxChr:  inc    si
[VU2]  187: ; Zeiger erhöhen
[TG2]  188:         jmp     ParseLop
[C35]  189: ; und Zeile weiter untersuchen
[ØØØ]  19Ø:
[KT1]  191: ChkOption:
[9X5]  192: ; Nachschauen, ob Option in Ordnung
[D51]  193:         inc     si
[VU2]  194: ; Zeiger erhöhen
[472]  195:         mov     al,[si]
[Z64]  196: ; und nächstes Byte lesen
[LF2]  197:         and     al,11Ø11111b
[TV5]  198: ; Klein- in Großbuchstaben wandeln
[ØM1]  199:         cmp     al,'L'
[892]  2ØØ: ; LIST Option?
[7M2]  2Ø1:         jnz     ChkOpt_W
[H34]  2Ø2:         mov     byte ptr LstFlag,Ø
[1Z2]  2Ø3: ; LIST Flag setzen
[MT1]  2Ø4:         jmp     NxChr
[ØØØ]  2Ø5:
[WN1]  2Ø6: ChkOpt_W:
[BM1]  2Ø7:         cmp     al,'W'
[NF2]  2Ø8: ; WRITE Option
[RH2]  2Ø9:         jnz     ChkOpt_C
[6W3]  21Ø:         mov     byte ptr WrFlag,Ø
[XA3]  211: ; WRITE Flag setzen
[MT1]  212:         jmp     NxChr
[ØØØ]  213:
[8N1]  214: ChkOpt_C:
[RL1]  215:         cmp     al,'C'
[MM2]  216: ; COMPARE Option
[DK2]  217:         jnz     CmdLinErr
```

```
[X14]  218:         mov     byte ptr CmpFlag,Ø
[SA3]  219: ; COMPARE Flag setzen
[MT1]  22Ø:         jmp     NxChr
[ØØØ]  221:
[XR1]  222: CmdLinErr:
[RP2]  223:         mov     ah,PrtStrg
[AU4]  224: ; Fehler-Meldung ausdrucken:
[FU3]  225:         mov     dx,offset CmdErr
[KH4]  226: ;  inkorrekte Kommando-Zeile
[UU1]  227:         int     msdos
[RP2]  228:         mov     ah,PrtStrg
[2S4]  229: ; korrekte Syntax ausdrucken
[WX3]  23Ø:         mov     dx,offset Syntax
[UU1]  231:         int     msdos
[ØK1]  232:         mov     al,1
[HX5]  233: ; Error-Code: Falsche Kommando-Zeile
[AX1]  234:         jmp     PgmEnd
[GZ2]  235: ; Programm beenden
[ØØØ]  236:
[EG1]  237: ChkFlags:
[GB6]  238: ; Nachsehen, ob ein oder mehrere Flags
[VA2]  239: ; gesetzt sind
[BZ3]  24Ø:         mov     al,byte ptr LstFlag
[BN3]  241:         and     al,byte ptr WrFlag
[7V3]  242:         and     al,byte ptr CmpFlag
[G82]  243:         jz      BuildTab
[XP4]  244: ; Weiter, falls Flag gesetzt
[RP2]  245:         mov     ah,PrtStrg
[UX6]  246: ; sonst Hilfe ausgeben: korrekte Syntax
[WX3]  247:         mov     dx,offset Syntax
[UU1]  248:         int     msdos
[BT1]  249:         jmp     OkEnd
[RH4]  25Ø: ; Programm-Ende, kein Fehler
[ØØØ]  251:
[NF6]  252: ;=========================================
[ØØØ]  253:
[HH1]  254: BuildTab:
[TW4]  255: ; Neue Vektor-Tabelle aufbauen
[HY3]  256:         mov     word ptr NewCS,cs
[9G6]  257: ; Aktuelles Code-Segment als Index für
[6S4]  258: ; benutzten Speicher ablegen
[BX2]  259:         mov     al,FirstInt
[256]  26Ø: ; Nummer des ersten Interrupts laden
[RU3]  261:         mov     di,offset NewVecs
[VC5]  262: ; Adresse für Int-Vektor Puffer
[ØØØ]  263:
[VU1]  264: GetIntLop:
[4A1]  265:         push    ax
[TR4]  266: ; aktuelle Int-Nummer sichern
[991]  267:         push    di
[2M5]  268: ; Vektor-Puffer-Adresse sichern
[B33]  269:         mov     ah,GetIntVec
[UU1]  27Ø:         int     msdos
[NØ6]  271: ; Interrupt-Vektor nach ES:BX laden
[371]  272:         pop     di
[X57]  273: ; Vektor-Puffer-Adresse wieder herstellen
[Ø82]  274:         mov     [di],es
[8T2]  275: ; Vektor sichern
[TA2]  276:         mov     [di+2],bx
[8E1]  277:         add     di,4
[D64]  278: ; Puffer-Adresse erhöhen
[Y71]  279:         pop     ax
[W86]  28Ø: ; Interrupt-Nummer wiederherstellen
[151]  281:         inc     al
[AK3]  282: ; und um eins erhöhen
[WX2]  283:         cmp     al,LastInt+1
[VS5]  284: ; Letzten Int-Vektor schon geholt?
[XL2]  285:         jnz     GetIntLop
[B84]  286: ; Falls nicht, dann weiter
[ØØØ]  287:
[NF6]  288: ;=========================================
[ØØØ]  289:
[AS1]  29Ø: ListTable:
[XØ7]  291: ; Vektor-Liste ausgeben, falls gewünscht
[BZ3]  292:         mov     al,byte ptr LstFlag
[GS2]  293: ; LIST-Flag holen
[1G1]  294:         or      al,al
[Y14]  295: ; testen, ob Flag gesetzt
[2A3]  296:         jnz     CompareTables
[B84]  297: ; falls nicht, dann weiter
[RP2]  298:         mov     ah,PrtStrg
[EF4]  299: ; Tabellen-Kopf ausdrucken
[FT3]  3ØØ:         mov     dx,offset TabHead
[UU1]  3Ø1:         int     msdos
[TP3]  3Ø2:         mov     ax,word ptr NewCS
[A25]  3Ø3: ; aktuelles Code-Segment laden
[AZ1]  3Ø4:         call    PrtHex
[NT3]  3Ø5: ; als HEX-Zahl ausgeben
[RP2]  3Ø6:         mov     ah,PrtStrg
[7W3]  3Ø7:         mov     dx,offset TabCont
[TN3]  3Ø8: ; weiter Text ausgeben
[UU1]  3Ø9:         int     msdos
[CX2]  31Ø:         mov     bl,FirstInt
```

Listing 1. Mit »chkints.asm« haben Sie die Interrupt-Vektoren fest im Griff

```
[ØN5]  311: ; Interrupt-Index initialisieren
[6V3]  312:         mov    si,offset NewVecs
[U95]  313: ; Vektor-Zeiger initialisieren
[ØØØ]  314:
[KG1]  315: ListLop:
[SX5]  316: ; Alle Vektoren in Schleife ausgeben
[5A1]  317:         push   bx
[474]  318: ; Interrupt-Index retten
[S91]  319:         push   si
[U95]  32Ø: ; Vektor-Zeiger initialisieren
[RP2]  321:         mov    ah,PrtStrg
[L95]  322: ; ersten Teil der Zeile drucken
[7E3]  323:         mov    dx,offset Part1
[TK1]  324: ; 'Adr(Int '
[UU1]  325:         int    msdos
[K71]  326:         pop    si
[Z71]  327:         pop    bx
[W86]  328: ; Interrupt-Nummer wiederherstellen
[5A1]  329:         push   bx
[D34]  33Ø: ; und alles wieder retten
[S91]  331:         push   si
[GP1]  332:         mov    al,bl
[K25]  333: ; Interrupt-Nummer ausdrucken
[KR2]  334:         call   PrtHexByte
[RP2]  335:         mov    ah,PrtStrg
[Ø85]  336: ; zweiten Teil der Zeile drucken
[TTØ]  337: ; ') = '
[AE3]  338:         mov    dx,offset Part2
[UU1]  339:         int    msdos
[K71]  34Ø:         pop    si
[MV3]  341: ; Vektor-Pointer holen
[S91]  342:         push   si
[PA3]  343: ; und wieder sichern
[FG2]  344:         call   PrtHexAdr
[2K5]  345: ; Vektor ausgeben: Segment:Offset
[RP2]  346:         mov    ah,PrtStrg
[C65]  347: ; dritten Teil der Zeile drucken
[F93]  348:         mov    dx,offset crlf
[UU1]  349:         int    msdos
[K71]  35Ø:         pop    si
[LR5]  351: ; Vektor-Zeiger wiederherstellen
[RE1]  352:         add    si,4
[VW6]  353: ; und für nächsten Durchlauf vorbereiten
[Z71]  354:         pop    bx
[Z24]  355: ; Interrupt-Index holen
[251]  356:         inc    bl
[XX2]  357:         cmp    bl,LastInt+1
[K93]  358: ; alles ausgegeben?
[L62]  359:         jne    ListLop
[356]  36Ø: ; falls nicht, dann nächste Schleife
[RP2]  361:         mov    ah,PrtStrg
[9G4]  362: ; Zeilenvorschub anhängen
[F93]  363:         mov    dx,offset crlf
[UU1]  364:         int    msdos
[ØØØ]  365:
[NF6]  366: ;=======================================
[ØØØ]  367:
[8F2]  368: CompareTables:
[6X5]  369: ; Tabellen vergleichen, falls nötig
[AY3]  37Ø:         mov    al,byte ptr CmpFlag
[943]  371: ; COMPARE-Flag holen
[1G1]  372:         or     al,al
[Y14]  373: ; testen, ob Flag gesetzt
[6C2]  374:         jz     DoCompare
[FN2]  375:         jmp    WriteTable
[B84]  376: ; falls nicht, dann weiter
[ØØØ]  377:
[8R1]  378: DoCompare:
[PT2]  379:         mov    ah,OpHandle
[HA3]  38Ø: ; Alte Datei einlesen
[ZX2]  381: ; zuerst öffnen
[WH1]  382:         mov    al,Ø
[MK3]  383: ; Zugriffscode: Lesen
[UF3]  384:         mov    dx,offset Fname
[N56]  385: ; Dateiname im aktuellen Verzeichnis!
[UU1]  386:         int    msdos
[D22]  387:         jnc    RdTable
[V16]  388: ; Falls kein Fehler, dann alte Tabelle
[ØØØ]  389:
[TT1]  39Ø: ;  einlesen
[KG1]  391:         cmp    al,2
[DW3]  392: ; Datei nicht gefunden?
[VP1]  393:         je     NoFile
[1A3]  394: ; Dann Bescheid sagen
[SH2]  395:         jmp    FileError
[GD5]  396: ; Verzweige, falls anderer Fehler
[ØØØ]  397:
[T91]  398: NoFile:
[RP2]  399:         mov    ah,PrtStrg
[963]  4ØØ: ; Meldung ausgeben:
[575]  4Ø1: ; 'Datei INTS.ADR nicht gefunden'
[484]  4Ø2:         mov    dx,offset NoFileMsg
[UU1]  4Ø3:         int    msdos
[4K1]  4Ø4:         mov    al,2
[355]  4Ø5: ; Fehlercode wieder herstellen
[AX1]  4Ø6:         jmp    PgmEnd
[7Y3]  4Ø7: ; und Programm abbrechen
[ØØØ]  4Ø8:
[RD1]  4Ø9: RdTable:
[NØ4]  41Ø:         mov    word ptr Handle,ax
[EF7]  411: ; Kanal-Nummer der geöffneten Datei sichern
[HR1]  412:         mov    bx,ax
[F33]  413: ;  und in BX kopieren
[PT2]  414:         mov    ah,RdHandle
[AX4]  415: ; alte Vektor-Tabelle einlesen
[SC3]  416:         mov    cx,OldCS-NewCS
[MW3]  417: ; Zahl zu lesender Bytes
[6H3]  418:         mov    dx,offset OldCS
[T62]  419: ; Zieladresse
[UU1]  42Ø:         int    msdos
[GM1]  421:         jnc    ok_1
[CY3]  422: ; weiter, falls alles ok
[SH2]  423:         jmp    FileError
[766]  424: ; Verzweige, falls Fehler aufgetreten
[ØØØ]  425:
[ØXØ]  426: ok_1:
[2S2]  427:         mov    ah,ClHandle
[WB4]  428: ; Datei wieder schließen
[ST3]  429:         mov    bx,word ptr Handle
[AT4]  43Ø: ; Kanal-Nummer zurückholen
[UU1]  431:         int    msdos
[642]  432:         jnc    Compare
[WV6]  433: ; weiter, falls erfolgreich geschlossen
[SH2]  434:         jmp    FileError
[4V6]  435: ; sonst wieder Fehlermeldung und Abbruch
[ØØØ]  436:
[XD1]  437: Compare:
[RP2]  438:         mov    ah,PrtStrg
[226]  439: ; Kopfzeile für Vergleich ausgeben
[XX3]  44Ø:         mov    dx,offset DiffHead
[UU1]  441:         int    msdos
[B54]  442:         mov    byte ptr DiffFlag,Ø
[ZY5]  443: ; DiffFlag zurücksetzen: noch kein
[BT3]  444: ;  Unterschied gefunden
[TP3]  445:         mov    ax,word ptr NewCS
[W86]  446: ; Zuerst werden altes und neues Code-
[NC3]  447: ; Segment verglichen
[XL3]  448:         cmp    ax,word ptr OldCS
[CC2]  449:         je     Compare_1
[NF2]  45Ø:         call   PrtCsDiff
[RP3]  451: ; Unterschied ausgeben
[ØØØ]  452:
[KR1]  453: Compare_1:
[CX2]  454:         mov    bl,FirstInt
[UZ5]  455: ; Interrupt-Zähler initialisieren
[6V3]  456:         mov    si,offset NewVecs
[U95]  457: ; Vektor-Zeiger initialisieren
[3V3]  458:         mov    di,offset OldVecs
[ØØØ]  459:
[7B2]  46Ø: Compare_Loop:
[W72]  461:         mov    ax,[si]
[LV4]  462: ; Zuerst Segment vergleichen
[V52]  463:         cmp    ax,[di]
[7N2]  464:         jne    PrtVecDiff
[RV6]  465: ; falls nicht identisch, dann Unterschied
[6T1]  466: ;  ausgeben
[BC2]  467:         mov    ax,[si+2]
[ND4]  468: ; Jetzt Offset vergleichen
[AA2]  469:         cmp    ax,[di+2]
[7N2]  47Ø:         jne    PrtVecDiff
[ØØØ]  471:
[AM1]  472: NextComp:
[RE1]  473:         add    si,4
[VU2]  474: ; Zeiger erhöhen
[8E1]  475:         add    di,4
[251]  476:         inc    bl
[VØ5]  477: ; Interrupt-Zähler erhöhen
[XX2]  478:         cmp    bl,LastInt+1
[ZØ3]  479: ; Schleife bis Ende
[193]  48Ø:         jne    Compare_Loop
[ØØØ]  481:
[RP2]  482:         mov    ah,PrtStrg
[LB4]  483: ; Zeilenvorschub ausgeben
[F93]  484:         mov    dx,offset crlf
[UU1]  485:         int    msdos
[Z54]  486:         mov    al,byte ptr DiffFlag
[V56]  487: ; irgendwelche Unterschiede gefunden?
[1G1]  488:         or     al,al
[CP2]  489:         jnz    CompareEnd
[685]  49Ø: ; Programm fortsetzen, falls ja
[RP2]  491:         mov    ah,PrtStrg
[FG6]  492: ; sonst zuerst Meldung 'keine' ausgeben
[CT3]  493:         mov    dx,offset NoDiff
[UU1]  494:         int    msdos
[ØØØ]  495:
[SØ2]  496: CompareEnd:
[FN2]  497:         jmp    WriteTable
[B35]  498: ; und dann Programm fortsetzen
[ØØØ]  499:
[E12]  5ØØ: PrtVecDiff:
[6L5]  5Ø1: ; alten und neuen Vektor ausgeben
```

```
[TM4]  502:          mov     byte ptr DiffFlag,0ffh
[TW4]  503: ; DiffFlag setzen: Unterschied
[W34]  504:          mov     byte ptr IntNum,bl
[595]  505: ; alle Zähler und Zeiger retten
[134]  506:          mov     word ptr Source,si
[XV3]  507:          mov     word ptr Dest,di
[RP2]  508:          mov     ah,PrtStrg
[6F4]  509: ; Beginn der Zeile ausgeben
[0B2]  510: ; 'Interrupt: '
[9R3]  511:          mov     dx,offset Diff_1
[UU1]  512:          int     msdos
[AU3]  513:          mov     al,byte ptr IntNum
[WU4]  514: ; Interrupt-Nummer ausgeben
[KR2]  515:          call    PrtHexByte
[RP2]  516:          mov     ah,PrtStrg
[YB5]  517: ; zweiten Teil der Zeile ausgeben
[P53]  518: ; 'h:  Alter Vektor: '
[DR3]  519:          mov     dx,offset Diff_2
[UU1]  520:          int     msdos
[8N3]  521:          mov     si,word ptr Dest
[N26]  522: ; Adresse des alten Vektors ausgeben
[FG2]  523:          call    PrtHexAdr
[RP2]  524:          mov     ah,PrtStrg
[AA5]  525: ; dritten Teil der Zeile ausgeben
[AW2]  526: ; ', Neuer Vektor: '
[HR3]  527:          mov     dx,offset Diff_3
[UU1]  528:          int     msdos
[PW3]  529:          mov     si,word ptr Source
[446]  530: ; Adresse des neuen Vektors ausgeben
[FG2]  531:          call    PrtHexAdr
[RP2]  532:          mov     ah,PrtStrg
[RV3]  533: ; Ende der Zeile ausgeben
[F93]  534:          mov     dx,offset crlf
[UU1]  535:          int     msdos
[AU3]  536:          mov     al,byte ptr IntNum
[AP5]  537: ; Funktion des Interrupts ausgeben
[LA3]  538:          call    PrIntFunction
[TM3]  539:          mov     di,word ptr Dest
[7D6]  540: ; Zähler und Zeiger wiederherstellen
[PW3]  541:          mov     si,word ptr Source
[BU3]  542:          mov     bl,byte ptr IntNum
[LH2]  543:          jmp     NextComp
[TY4]  544: ; nächsten Vektor vergleichen
[000]  545:
[ZT1]  546: PrtCsDiff:
[NW4]  547: ; muß noch geschrieben werden
[TM4]  548:          mov     byte ptr DiffFlag,0ffh
[DF6]  549: ; DiffFlag setzen: Unterschied gefunden
[RP2]  550:          mov     ah,PrtStrg
[UK5]  551: ; aktuelles Code-Segment ausgeben
[Y14]  552:          mov     dx,offset NonActCs
[UU1]  553:          int     msdos
[FN3]  554:          mov     ax,word ptr OldCS
[AZ1]  555:          call    PrtHex
[RP2]  556:          mov     ah,PrtStrg
[4V4]  557: ; altes Code-Segment ausgeben
[8F3]  558:          mov     dx,offset ActCs
[UU1]  559:          int     msdos
[TP3]  560:          mov     ax,word ptr NewCS
[AZ1]  561:          call    PrtHex
[RP2]  562:          mov     ah,PrtStrg
[4V3]  563: ; 'Unterschied in Bytes:'
[G04]  564:          mov     dx,offset ByteDiff
[UU1]  565:          int     msdos
[TP3]  566:          mov     ax,word ptr NewCS
[GZ3]  567: ; Unterschied berechnen
[XL3]  568:          sub     ax,word ptr OldCS
[V22]  569:          jge     PosDiff
[S51]  570:          neg     ax
[LZ5]  571: ; negieren, falls Differenz negativ
[000]  572:
[LE1]  573: PosDiff:
[FR2]  574:          cmp     ax,0ffffh/16
[RL5]  575: ; Differenz größer als 64 KByte?
[7R1]  576:          jg      TooBig
[L56]  577: ; dann entsprechende Meldung ausgeben
[GM1]  578:          mov     cx,16
[W15]  579: ; Differenz in Bytes umrechnen
[W71]  580:          mul     cx
[GX1]  581:          call    PrtDec
[V45]  582: ; und als Dezimal-Zahl ausgeben
[RP2]  583:          mov     ah,PrtStrg
[3W2]  584: ; Zeilenvorschub
[F93]  585:          mov     dx,offset crlf
[UU1]  586:          int     msdos
[3P0]  587:          ret
[000]  588:
[T24]  589: TooBig: mov     ah,PrtStrg
[YL4]  590: ; Meldung: 'Mehr als 64 KByte'
[G84]  591:          mov     dx,offset TooBigMsg
[UU1]  592:          int     msdos
[3P0]  593:          ret
[000]  594:
[5U1]  595: FileError:
[4A1]  596:          push    ax
```

```
[FC3]  597: ; Fehler-Code sichern
[RP2]  598:          mov     ah,PrtStrg
[454]  599: ; Fehler-Meldung ausgeben
[AH4]  600:          mov     dx,offset FileErrMsg
[UU1]  601:          int     msdos
[Y71]  602:          pop     ax
[ED4]  603: ; Fehler-Code zurückholen
[AX1]  604:          jmp     PgmEnd
[000]  605:
[NF6]  606: ;=======================================
[000]  607:
[R12]  608: WriteTable:
[R16]  609: ; Neue Tabelle auf Diskette schreiben,
[5D2]  610: ; falls nötig
[ES3]  611:          mov     al,byte ptr WrFlag
[213]  612: ; WRITE-Flag holen
[1G1]  613:          or      al,al
[Y14]  614: ; testen, ob Flag gesetzt
[PU1]  615:          jnz     OkEnd
[ZU3]  616: ; falls nicht, dann Ende
[8S2]  617:          mov     ah,CrHandle
[485]  618: ; Datei 'INTS.ADR' anlegen/öffnen
[PK1]  619:          mov     cx,0
[1X2]  620: ; keine Attribute
[UF3]  621:          mov     dx,offset Fname
[UU1]  622:          int     msdos
[UA2]  623:          jc      FileError
[D26]  624: ; Falls Datei-Fehler, dann verzweigen
[N04]  625:          mov     word ptr Handle,ax
[855]  626: ; Kanal-Nummer der Datei merken
[CV1]  627:          mov     bx,ax
[4V6]  628: ; Kanal-Nummer in BX für Schreib-Aufruf
[EZ2]  629:          mov     ah,WrHandle
[XD3]  630:          mov     cx,OldCS-NewCS
[XP5]  631: ; Anzahl der zu schreibenden Bytes
[5S3]  632:          mov     dx,offset NewCS
[LD3]  633: ; Beginn Quellpuffer
[4Y1]  634:          int     msdos
[UL4]  635: ; Daten auf Platte schreiben
[ME2]  636:          jc      FileError
[D26]  637: ; Falls Datei-Fehler, dann verzweigen
[EY2]  638:          mov     ah,ClHandle
[WB4]  639: ; Datei wieder schließen
[G64]  640:          mov     bx,word ptr Handle
[4Y1]  641:          int     msdos
[ME2]  642:          jc      FileError
[D26]  643: ; Falls Datei-Fehler, dann verzweigen
[KU2]  644:          mov     ah,PrtStrg
[4B4]  645: ; Erfolgs-Meldung ausgeben
[UZ3]  646:          mov     dx,offset WriteOk
[4Y1]  647:          int     msdos
[4W1]  648:          jmp     OkEnd
[000]  649:
[NF6]  650: ;=======================================
[000]  651:
[WT2]  652: OkEnd:  mov     al,0
[N64]  653: ; Programm-Ende: No Error
[000]  654:
[4W3]  655: PgmEnd: mov     ah,RetDos
[D95]  656: ; Programm-Ende, Error-Code in AL
[4Y1]  657:          int     msdos
[000]  658:
[NF6]  659: ;=======================================
[000]  660:
[CN5]  661: ; PrtHex: Drucke Inhalt von AX als
[S53]  662: ;          16-Bit Hex-Zahl
[000]  663:
[L23]  664: PrtHex: mov     bl,4
[0N4]  665: ; 4 Nibbles sind auszugeben
[000]  666:
[DW1]  667: PrtHexLop:
[XF1]  668:          push    ax
[7Y1]  669: ; ax sichern
[ZF1]  670:          push    bx
[UA4]  671: ; Nibble-Zähler sichern
[DM1]  672:          mov     cl,4
[BS1]  673:          shr     ah,cl
[YS5]  674: ; höchstwertiges Nibble maskieren
[8N1]  675:          cmp     ah,10
[EA6]  676: ; falls >=10, dann Offset für Buchstaben
[6Y3]  677: ; addieren, sonst Ziffer
[0B2]  678:          jl      NoLetter
[CE2]  679:          add     ah,'A'-10-'0'
[000]  680:
[WL1]  681: NoLetter:
[4P1]  682:          add     ah,'0'
[9D6]  683: ; Ok, jetzt ist das Nibble in ein ASCII-
[7T4]  684: ; Zeichen umgewandelt worden
[CT1]  685:          mov     dl,ah
[ER4]  686: ; Zeichen via MS-DOS ausgeben
[HH2]  687:          mov     ah,PrtChr
[4Y1]  688:          int     msdos
```

Listing 1. Mit »chkints.asm« haben Sie die Interrupt-Vektoren fest im Griff

```
[TD1]  689:          pop     bx
[3X5]  690: ; Nibble-Zähler wiederherstellen
[RD1]  691:          pop     ax
[YK6]  692: ; auszugebendes Wort wiederherstellen
[DM1]  693:          mov     cl,4
[3T1]  694:          shl     ax,cl
[ER7]  695: ; nächstes Nibble in höchstwertige Position
[B91]  696:          dec     bl
[RE3]  697: ; Zähler erniedrigen
[5T2]  698:          jnz     PrtHexLop
[3PØ]  699:          ret
[ØØØ]  7ØØ:
[Z16]  7Ø1: ; PrtexByte: Drucke Inhalt von AL als
[MA3]  7Ø2: ;            8-Bit Hex-Zahl
[ØØØ]  7Ø3:
[Ø42]  7Ø4: PrtHexByte:
[2T1]  7Ø5:          mov     ah,al
[9M1]  7Ø6:          mov     bl,2
[HY3]  7Ø7: ; Nur zwei Nibble drucken
[UR2]  7Ø8:          jmp     PrtHexLop
[ØØØ]  7Ø9:
[A77]  71Ø: ; PrtHexAdr: Drucke das Doppelwort ab [si]
[3U5]  711: ;            in der Form Segment:Offset
[ØØØ]  712:
[UU1]  713: PrtHexAdr:
[9E2]  714:          mov     ax,[si]
[744]  715: ; SI zeigt auf Doppelwort
[MF1]  716:          push    si
[562]  717:          call    PrtHex
[4Z2]  718: ; Segment ausgeben
[HH2]  719:          mov     ah,PrtChr
[DR3]  72Ø: ; Doppelpunkt ausgeben
[BV1]  721:          mov     dl,':'
[4Y1]  722:          int     msdos
[ED1]  723:          pop     si
[NT2]  724: ; Offset ausgeben
[DF2]  725:          mov     ax,[si+2]
[562]  726:          call    PrtHex
[3PØ]  727:          ret
[ØØØ]  728:
[346]  729: ; PrtDec: AX als Dezimalzahl ausgeben,
[PZ5]  73Ø: ;         führende Nullen unterdrücken
[ØØØ]  731:
[LG3]  732: PrtDec: mov     cx,1ØØØØ
[N55]  733: ; CX enthält aktuelle Dezimale
[7M1]  734:          mov     bl,Ø
[HW5]  735: ; Druck-Flag für führende Nullen
[ØØØ]  736:
[K32]  737: PrtDecLoop:
[FN1]  738:          mov     dx,Ø
[6V4]  739: ; 16-Bit Division präparieren
[9D1]  74Ø:          div     cx
[UA4]  741: ; AX := DX.AX / CX, DX := Rest
[3G1]  742:          push    dx
[CV3]  743: ; Rest auf Stack sichern
[XR1]  744:          cmp     al,bl
[KM2]  745: ; führende Null?
[ZN2]  746:          je      ContPrtDec
[K93]  747: ; dann nichts drucken
[582]  748:          mov     bl,Øffh
[584]  749: ; sonst Druck-Flag setzen
[ZF1]  75Ø:          push    bx
[N22]  751: ; und sichern
[1G1]  752:          push    cx
[7M4]  753: ; aktuelle Dezimale sichern
[GP1]  754:          add     al,'Ø'
[EY5]  755: ; Zahl in al in ASCII-Zeichen wandeln
[MT1]  756:          mov     dl,al
[V26]  757: ; und als Funktions-Parameter nach dl
[HH2]  758:          mov     ah,PrtChr
[M95]  759: ; einzelnes Zeichen ausdrucken
[4Y1]  76Ø:          int     msdos
[VD1]  761:          pop     cx
[6N4]  762: ; Dezimale wiederherstellen
[TD1]  763:          pop     bx
[884]  764: ; Druck-Flag zurückholen
[ØØØ]  765:
[432]  766: ContPrtDec:
[CV1]  767:          mov     ax,cx
[FT6]  768: ; Dezimale verschieben, dazu zuerst nach
[5G5]  769: ; AX laden und dann durch 1Ø teilen
[ES1]  77Ø:          mov     cx,1Ø
[FN1]  771:          mov     dx,Ø
[9D1]  772:          div     cx
[BY3]  773: ; Division durchführen
[EV1]  774:          mov     cx,ax
[NU3]  775: ; Ergebnis wieder nach CX
[RD1]  776:          pop     ax
[325]  777: ; alten Divisions-Rest nach AX
[WL1]  778:          cmp     cx,1
[2B4]  779: ; letzte Dezimale drucken?
[DC2]  78Ø:          jne     TestEnd
[515]  781: ; falls nicht, dann schon Ende?
[582]  782:          mov     bl,Øffh
[L96]  783: ; spätestens bei der letzten Dezimale
[P17]  784: ; muß das Druck-Flag eingeschaltet werden!
[5X2]  785:          jmp     PrtDecLoop
[ØØØ]  786:
[NE1]  787: TestEnd:
[5N1]  788:          or      cx,cx
[E74]  789: ; alle Dezimalen gedruckt?
[GY2]  79Ø:          jnz     PrtDecLoop
[B84]  791: ; falls nicht, dann weiter
[3PØ]  792:          ret
[Z12]  793: ; sonst Ende
[ØØØ]  794:
[NF6]  795: ;=======================================
[ØØØ]  796:
[GN7]  797: ; PrIntFunction: Bedeutung eines Interrupts
[ØK5]  798: ;                ausgeben (falls möglich)
[P3Ø]  799: ;
[HV3]  8ØØ: ;  al: Interrupt-Nummer
[ØØØ]  8Ø1:
[3L2]  8Ø2: PrIntFunction:
[ØX1]  8Ø3:          cmp     al,77h
[LY3]  8Ø4: ; Interrupt-Nummer > 77h
[M72]  8Ø5:          jg      EndTest
[ER1]  8Ø6: ; dann Ende
[TL1]  8Ø7:          mov     ah,Ø
[U66]  8Ø8: ; sonst Vektor für Beschreibung des
[RT3]  8Ø9: ; Interrupts berechnen
[9R1]  81Ø:          add     ax,ax
[UM3]  811:          add     ax,offset IntTab
[ØØØ]  812:
[NG1]  813: PrintIt:
[U96]  814: ; ok, Adresse des String-Vektors in AX
[XF1]  815:          push    ax
[FS2]  816: ; Adresse sichern
[KU2]  817:          mov     ah,PrtStrg
[9D6]  818: ; zuerst ein paar Leerzeichen ausgeben
[4S3]  819:          mov     dx,offset blanks
[4Y1]  82Ø:          int     msdos
[KU2]  821:          mov     ah,PrtStrg
[ED1]  822:          pop     si
[CS4]  823: ; Jetzt den String ausgeben
[FE2]  824:          mov     dx,[si]
[4Y1]  825:          int     msdos
[ØØØ]  826:
[5F1]  827: EndTest:
[3PØ]  828:          ret
[ØØØ]  829:
[ØØØ]  83Ø:
[H87]  831: ; Vektoren zu den Interrupt-Beschreibungen
[ØØØ]  832:
[1U5]  833: IntTab:   dw intØØ, intØ1, intØ2, intØ3
[794]  834:           dw intØ4, intØ5, intØ6, intØ7
[KB4]  835:           dw intØ8, intØ9, intØa, intØb
[XD4]  836:           dw intØc, intØd, intØe, intØf
[984]  837:           dw int1Ø, int11, int12, int13
[H94]  838:           dw int14, int15, int16, int17
[WB4]  839:           dw int18, int19, int1a, int1b
[7E4]  84Ø:           dw int1c, int1d, int1e, int1f
[L84]  841:           dw int2Ø, int21, int22, int23
[V94]  842:           dw int24, int25, int26, int27
[6F3]  843:           dw (4Øh-28h) DUP(int28)
[794]  844:           dw int4Ø, int41, int42, int43
[FA4]  845:           dw int44, int45, int46, int47
[6A3]  846:           dw int48, int49, int4A
[6K3]  847:           dw (68h-4Bh) DUP(int4B)
[WF3]  848:           dw (7Øh-68h) DUP(int68)
[5A4]  849:           dw int7Ø, int71, int72, int73
[DB4]  85Ø:           dw int74, int75, int76, int77
[ØØØ]  851:
[BN4]  852: ; Interrupt-Beschreibungen
[ØØØ]  853:
[8T5]  854: intØØ:    db 'HW: Division durch Null'
[YV1]  855:           db cr,lf,etx
[BZ4]  856: intØ1:    db 'HW: Einzelschritt'
[YV1]  857:           db cr,lf,etx
[636]  858: intØ2:    db 'HW: NMI (z.B. Parity Error)'
[YV1]  859:           db cr,lf,etx
[BM7]  86Ø: intØ3:    db 'HW: Behandlung 1-Byte Softwar
       e-INT'
[YV1]  861:           db cr,lf,etx
[D67]  862: intØ4:    db 'HW: Überlauf / Behandlung INT
       O'
[YV1]  863:           db cr,lf,etx
[D57]  864: intØ5:    db 'HW: Print Screen / ab 8Ø186:
       BOUND'
[YV1]  865:           db cr,lf,etx
[81A]  866: intØ6:    db 'HW: Maskierbarer Interrupt /
       Ungültiger Op-Code'
[YV1]  867:           db cr,lf,etx
[9S6]  868: intØ7:    db 'HW: ab 8Ø286: kein Coprozesso
       r'
[YV1]  869:           db cr,lf,etx
[ØU7]  87Ø: intØ8:    db 'HW: Hardware-Timer-Interrupt:
        IRQØ'
[YV1]  871:           db cr,lf,etx
[2U6]  872: intØ9:    db 'HW: Tastatur-Interrupt: IRQ 1
       '
```

```
[YV1]  873:             db cr,lf,etx
[2B9]  874: intØa:      db 'HW: IRQ 2: 2ter Interrupt-Con
           troller beim AT'
[YV1]  875:             db cr,lf,etx
[GZ5]  876: intØb:      db 'HW: COM2-Interrupt (IRQ 3)'
[YV1]  877:             db cr,lf,etx
[LZ5]  878: intØc:      db 'HW: COM1-Interrupt (IRQ 4)'
[YV1]  879:             db cr,lf,etx
[F16]  88Ø: intØd:      db 'HW: LPT2-Interrupt (IRQ 5)'
[YV1]  881:             db cr,lf,etx
[Z56]  882: intØe:      db 'HW: Disk-Controller (IRQ 6)'
[YV1]  883:             db cr,lf,etx
[S16]  884: intØf:      db 'HW: LPT1-Interrupt (IRQ 7)'
[YV1]  885:             db cr,lf,etx
[XØ7]  886: int1Ø:      db 'BIOS: Bildschirm-Manipulation
           '
[YV1]  887:             db cr,lf,etx
[TE7]  888: int11:      db 'BIOS: Vorhandene Hardware mel
           den'
[YV1]  889:             db cr,lf,etx
[WF7]  89Ø: int12:      db 'BIOS: Speichergröße ermitteln
           '
[YV1]  891:             db cr,lf,etx
[WDA]  892: int13:      db 'BIOS: Behandlung Disketten- /
            Festplatten-Laufwerk'
[YV1]  893:             db cr,lf,etx
[AN9]  894: int14:      db 'BIOS: Behandlung der serielle
           n Schnittstellen'
[YV1]  895:             db cr,lf,etx
[YEA]  896: int15:      db 'BIOS: Special-Interest-Interr
           upt (je nach Rechner)'
[YV1]  897:             db cr,lf,etx
[YR5]  898: int16:      db 'BIOS: Tastatur-Abfrage'
[YV1]  899:             db cr,lf,etx
[PP5]  9ØØ: int17:      db 'BIOS: Drucker-Ausgabe'
[YV1]  9Ø1:             db cr,lf,etx
[PE7]  9Ø2: int18:      db 'ROM-BASIC (nur bei Original-I
           BMs)'
[YV1]  9Ø3:             db cr,lf,etx
[NM4]  9Ø4: int19:      db 'BIOS: Bootstrap'
[YV1]  9Ø5:             db cr,lf,etx
[XZ4]  9Ø6: int1a:      db 'BIOS: Systemzeit'
[YV1]  9Ø7:             db cr,lf,etx
[5C6]  9Ø8: int1b:      db 'BIOS/DOS: Ctrl-Brk-Vektor'
[YV1]  9Ø9:             db cr,lf,etx
[3T5]  91Ø: int1c:      db 'BIOS: Timer-Interrupt'
[YV1]  911:             db cr,lf,etx
[CØ8]  912: int1d:      db 'BIOS: Bildschirm-Parameter (A
           dresse)'
[YV1]  913:             db cr,lf,etx
[VM7]  914: int1e:      db 'BIOS/DOS: Disk-Parameter (Adr
           esse)'
[YV1]  915:             db cr,lf,etx
[HB8]  916: int1f:      db 'BIOS/DOS: Grafik-Zeichensatz
           (Adresse)'
[YV1]  917:             db cr,lf,etx
[XF5]  918: int2Ø:      db 'DOS: Programm beenden'
[YV1]  919:             db cr,lf,etx
[9M5]  92Ø: int21:      db 'DOS-Funktionsaufrufe'
[YV1]  921:             db cr,lf,etx
[DG8]  922: int22:      db 'DOS: Programmende-Behandlung
           (Prozedur)'
[YV1]  923:             db cr,lf,etx
[ØØ8]  924: int23:      db 'DOS: Control-C-Behandlung (Pr
           ozedur)'
[YV1]  925:             db cr,lf,etx
[1E7]  926: int24:      db 'DOS: Fehler-Behandlung (Proze
           dur)'
[YV1]  927:             db cr,lf,etx
[ØN6]  928: int25:      db 'DOS: Diskette / Platte lesen'
[YV1]  929:             db cr,lf,etx
[XB7]  93Ø: int26:      db 'DOS: Diskette / Platte schrei
           ben'
[YV1]  931:             db cr,lf,etx
[CM9]  932: int27:      db 'DOS: Programmende, Programm b
           leibt im Speicher'
[YV1]  933:             db cr,lf,etx
[SKA]  934: int28:      db 'DOS: Reserviert für nicht dok
           umentierte Funktionen'
[YV1]  935:             db cr,lf,etx
[ØØØ]  936:
[SH6]  937: int4Ø:      db 'BIOS: Disketten-Funktionen'
[YV1]  938:             db cr,lf,etx
[LD8]  939: int41:      db 'Adresse der ersten Festplatte
           n-Tabelle'
[YV1]  94Ø:             db cr,lf,etx
[KØ1]  941: int42:
[MØ1]  942: int43:
[PØ1]  943: int44:
[AU3]  944: int45:      db 'Reserviert'
[YV1]  945:             db cr,lf,etx
[LP8]  946: int46:      db 'Adresse der zweiten Festplatt
           en-Tabelle'
[YV1]  947:             db cr,lf,etx
[WØ1]  948: int47:
[YØ1]  949: int48:
[ØG5]  95Ø: int49:      db 'Frei für Anwendungen'
```

```
[YV1]  951:             db cr,lf,etx
[6U5]  952: int4A:      db 'AT: Alarm-Zeit erreicht'
[YV1]  953:             db cr,lf,etx
[KG5]  954: int4B:      db 'Frei für Anwendungen'
[YV1]  955:             db cr,lf,etx
[ØØØ]  956:
[PS3]  957: int68:      db 'Unbenutzt'
[YV1]  958:             db cr,lf,etx
[ØØØ]  959:
[FG6]  96Ø: int7Ø:      db 'AT-BIOS: Echtzeituhr (IRQ 8)'
[YV1]  961:             db cr,lf,etx
[59A]  962: int71:      db 'AT-BIOS: Aufruf INTØA, dadurc
           h Simulation von IRQ2'
[YV1]  963:             db cr,lf,etx
[VY6]  964: int72:      db 'AT-BIOS: COM3-Interrupt (IRQ
           1Ø)'
[YV1]  965:             db cr,lf,etx
[3Z6]  966: int73:      db 'AT-BIOS: COM4-Interrupt (IRQ
           11)'
[YV1]  967:             db cr,lf,etx
[V76]  968: int74:      db 'AT-BIOS: Reserviert (IRQ 12)'
[YV1]  969:             db cr,lf,etx
[TM6]  97Ø: int75:      db 'AT-BIOS: Coprozessor (IRQ 13)
           '
[YV1]  971:             db cr,lf,etx
[BE8]  972: int76:      db 'AT-BIOS: Festplatten-Controll
           er (IRQ 14)'
[YV1]  973:             db cr,lf,etx
[786]  974: int77:      db 'AT-BIOS: Reserviert (IRQ 15)'
[YV1]  975:             db cr,lf,etx
[ØØØ]  976:
[NF6]  977: ;=======================================
[ØØØ]  978:
[1D3]  979: ; Beginn Datenbereich
[624]  98Ø: ; (Variablen und Strings)
[ØØØ]  981:
[7G2]  982: LstFlag:       db ?
[PW3]  983: ; Ø: ganze Liste ausgeben
[PC2]  984: WrFlag:        db ?
[675]  985: ; Ø: Liste auf Diskette schreiben
[LE2]  986: CmpFlag:       db ?
[LØ5]  987: ; Ø: Liste mit Datei vergleichen
[ØØØ]  988:
[ZE2]  989: DiffFlag:      db ?
[UV5]  99Ø: ; Ø: kein Unterschied zwischen alter
[ØØØ]  991:
[445]  992: ;  und neuer Int-Vektor-Tabelle
[ØØØ]  993:
[ST3]  994: Fname:         db 'INTS.ADR',Ø
[DL5]  995: ; Dateiname der Liste auf Diskette
[ØØØ]  996:
[PD6]  997: CmdErr:        db 'Fehler in Kommandozeile'
[NG2]  998:                 db cr,lf,lf,etx
[ØØØ]  999:
[5PA] 1ØØØ: Syntax: db 'Programm zur Untersuchung der I
           nterrupt-Vektoren'
[3S1] 1Ø01:         db cr,lf,lf
[479] 1Ø02:         db 'Syntax: CHKINTS     Hilfe: Korr
           ekte Syntax ausgeben'
[F51] 1Ø03:         db cr,lf
[HLB] 1Ø04:         db '        CHKINTS /L  aktuelle In
           terrupt-Liste auf Bildschirm ausgeben'
[F51] 1Ø05:         db cr,lf
[3TA] 1Ø06:         db '        CHKINTS /W  aktuelle In
           terrupt-Liste in Datei schreiben'
[F51] 1Ø07:         db cr,lf
[21B] 1Ø08:         db '        CHKINTS /C  aktuelle Li
           ste mit Datei vergleichen und Unter-'
[F51] 1Ø09:         db cr,lf
[XF7] 1Ø1Ø:         db '                    schiede auf
            Bildschirm ausgeben'
[3S1] 1Ø11:         db cr,lf,lf
[V38] 1Ø12:         db '    Alle Parameter dürfen auch
           gemischt werden'
[NG2] 1Ø13:         db cr,lf,lf,etx
[ØØØ] 1Ø14:
[TR6] 1Ø15: TabHead:    db ' Interrupt-Vektor-Tabelle:'
[F51] 1Ø16:             db cr,lf
[GRD] 1Ø17:             db '-------------------------'
[3S1] 1Ø18:             db cr,lf,lf
[5L4] 1Ø19:             db 'Aktuelles Code-Segment: '
[3YØ] 1Ø2Ø:             db etx
[VM5] 1Ø21: NonActCs:   db ' Altes Code-Segment: '
[3YØ] 1Ø22:             db etx
[UD6] 1Ø23: ActCs:      db 'h, aktuelles Code-Segment: '
[3YØ] 1Ø24:             db etx
[9E3] 1Ø25: ByteDiff:   db 'h',cr,lf
[3N4] 1Ø26:             db '    Unterschied in Bytes: '
[3YØ] 1Ø27:             db etx
[LD4] 1Ø28: TabCont:    db 'h',cr,lf,lf,etx
[BC4] 1Ø29: TooBigMsg:  db 'mehr als 64 K'
[YV1] 1Ø3Ø:             db cr,lf,etx
[ØØØ] 1Ø31:
```

Listing 1. Mit »chkints.asm« haben Sie die Interrupt-Vektoren fest im Griff

```
[9U3] 1Ø32: Part1:      db 'Adr(Int ',etx
[B13] 1Ø33: Part2:      db ') = ',etx
[B33] 1Ø34: blanks:     db '    ',etx
[KB3] 1Ø35: crlf:       db cr,lf,etx
[ØØØ] 1Ø36:
[8Y6] 1Ø37: DiffHead:   db 'Unterschiede zwischen alter'
[G44] 1Ø38:             db ' und neuer Interrupt-'
[EU4] 1Ø39:             db 'Vektor-Tabelle:',cr,lf,etx
[TT4] 1Ø4Ø: Diff_1:     db ' Interrupt: ',etx
[AF5] 1Ø41: Diff_2:     db 'h:  Alter Vektor: ',etx
[345] 1Ø42: Diff_3      db ', neuer Vektor: ',etx
[555] 1Ø43: NoDiff:     db ' Keine!',cr,lf,lf,etx
[ØØØ] 1Ø44:
[HE9] 1Ø45: WriteOk:    db 'Neue Datei INTS.ADR erfolgr
          eich erstellt!'
[NG2] 1Ø46:             db cr,lf,lf,etx
[L37] 1Ø47: FileErrMsg: db 'Fehler beim Datei-Handling'
[NG2] 1Ø48:             db cr,lf,lf,etx
[1RB] 1Ø49: NoFileMsg:  db 'Datei INTS.ADR im aktuellen
           Verzeichnis nicht gefunden'
[NG2] 1Ø5Ø:             db cr,lf,lf,etx
[ØØØ] 1Ø51:
[4Ø2] 1Ø52: NewCS:    dw ?
[686] 1Ø53: ; Puffer für aktuelles Code-Segment
[DZ6] 1Ø54: NewVecs:  dw 2*(LastInt-FirstInt+1) DUP (?)
[VG4] 1Ø55: ; Puffer für INT-Vektoren
[ØØØ] 1Ø56:
[DY1] 1Ø57: OldCS:    dw ?
[NG5] 1Ø58: ; Puffer für altes Code-Segment
[SX6] 1Ø59: OldVecs:  dw 2*(LastInt-FirstInt+1) DUP (?)
[VG4] 1Ø6Ø: ; Puffer für INT-Vektoren
[ØØØ] 1Ø61:
[222] 1Ø62: Handle:   dw ?
[ØØØ] 1Ø63:
[252] 1Ø64: IntNum:   db ?
[U42] 1Ø65: Source:   dw ?
[FV1] 1Ø66: Dest:     dw ?
[ØØØ] 1Ø67:
[YL1] 1Ø68:             db 1ØØ DUP(?)
[953] 1Ø69: ; Platz für Stack
[GK1] 1Ø7Ø: NewStack:
[ØØØ] 1Ø71:
[XW1] 1Ø72: cseg        ends
[R12] 1Ø73:             end  ChkInts
```

Listing 1. Mit »chkints.asm« haben Sie die Interrupt-Vektoren fest im Griff

```
[3K5]    1: EB 37 9Ø ØD ØA 43 48 4B 49 4E 54 53 2Ø 76 65 72
[885]    2: 73 2Ø 31 2E 33 2Ø 2Ø 2Ø 2Ø 2Ø 28 43 29 2Ø 31 39
[VC5]    3: 38 38 2Ø 2Ø 44 4F 53 2Ø 49 6E 74 65 72 6E 61 74
[WR5]    4: 69 6F 6E 61 6C ØD ØA ØA 24 BC BC 14 B4 4A BB 4C
[PL5]    5: Ø1 9Ø CD 21 B4 Ø9 BA Ø3 Ø1 CD 21 C6 Ø6 B8 ØC FF
[5U5]    6: C6 Ø6 B9 ØC FF C6 Ø6 BA ØC FF BE 81 ØØ 8A Ø4 3C
[YM5]    7: 2Ø 74 ØF 3C Ø9 74 ØB 3C 2F 74 ØA 3C ØD 74 3F EB
[CN5]    8: 2A 9Ø 46 EB E8 46 8A Ø4 24 DF 3C 4C 75 Ø7 C6 Ø6
[PS5]    9: B8 ØC ØØ EB ED 3C 57 75 Ø7 C6 Ø6 B9 ØC ØØ EB E2
[HS5]   1Ø: 3C 43 75 Ø7 C6 Ø6 BA ØC ØØ EB D7 B4 Ø9 BA C5 ØC
[AR5]   11: CD 21 B4 Ø9 BA EØ ØC CD 21 BØ Ø1 E9 15 Ø2 AØ B8
[CP5]   12: ØC 22 Ø6 B9 ØC 22 Ø6 BA ØC 74 ØA B4 Ø9 BA EØ ØC
[ZR5]   13: CD 21 E9 FC Ø1 8C ØE 4D 1Ø BØ ØØ BF 4F 1Ø 5Ø 57
[BP5]   14: B4 35 CD 21 5F 8C Ø5 89 5D Ø2 83 C7 Ø4 58 FE CØ
[AP5]   15: 3C 8Ø 75 EA AØ B8 ØC ØA CØ 75 51 B4 Ø9 BA 7C ØE
[KT5]   16: CD 21 A1 4D 1Ø E8 CF Ø1 B4 Ø9 BA 1F ØF CD 21 B3
[HK5]   17: ØØ BE 4F 1Ø 53 56 B4 Ø9 BA 34 ØF CD 21 5E 5B 53
[DU5]   18: 56 8A C3 E8 D5 Ø1 B4 Ø9 BA 3D ØF CD 21 5E 56 E8
[LU5]   19: CF Ø1 B4 Ø9 BA 47 ØF CD 21 5E 83 C6 Ø4 5B FE C3
[SP5]   2Ø: 8Ø FB 8Ø 75 CF B4 Ø9 BA 47 ØF CD 21 AØ BA ØC ØA
[DP5]   21: CØ 74 Ø3 E9 42 Ø1 B4 3D BØ ØØ BA BC ØC CD 21 73
[1D5]   22: 13 3C Ø2 74 Ø3 E9 24 Ø1 B4 Ø9 BA 13 1Ø CD 21 BØ
[LK5]   23: Ø2 E9 5F Ø1 A3 51 14 8B D8 B4 3F B9 Ø2 Ø2 9Ø BA
[PM5]   24: 4F 12 CD 21 73 Ø3 E9 Ø3 Ø1 B4 3E 8B 1E 51 14 CD
[EP5]   25: 21 73 Ø3 E9 F6 ØØ B4 Ø9 BA 4A ØF CD 21 C6 Ø6 BB
[3H5]   26: ØC ØØ A1 4D 1Ø 3B Ø6 4F 12 74 Ø3 E8 9Ø ØØ B3 ØØ
[VN5]   27: BE 4F 1Ø BF 51 12 8B Ø4 3B Ø5 75 2D 8B 44 Ø2 3B
[YG5]   28: 45 Ø2 75 25 83 C6 Ø4 83 C7 Ø4 FE C3 8Ø FB 8Ø 75
[WM5]   29: E5 B4 Ø9 BA 47 ØF CD 21 AØ BB ØC ØA CØ 75 Ø7 B4
[EY5]   3Ø: Ø9 BA BD ØF CD 21 E9 AF ØØ C6 Ø6 BB ØC FF 88 1E
[4N5]   31: 53 14 89 36 54 14 89 3E 56 14 B4 Ø9 BA 8C ØF CD
[NK5]   32: 21 AØ 53 14 E8 F4 ØØ B4 Ø9 BA 99 ØF CD 21 8B 36
[UR5]   33: 56 14 E8 EC ØØ B4 Ø9 BA AC ØF CD 21 8B 36 54 14
[SL5]   34: E8 DE ØØ B4 Ø9 BA 47 ØF CD 21 AØ 53 14 E8 1D Ø1
[PL5]   35: 8B 3E 56 14 8B 36 54 14 8A 1E 53 14 EB 86 C6 Ø6
[WV5]   36: BB ØC FF B4 Ø9 BA CF ØE CD 21 A1 4F 12 E8 87 ØØ
[ZS5]   37: B4 Ø9 BA E5 ØE CD 21 A1 4D 1Ø E8 7A ØØ B4 Ø9 BA
[4N5]   38: Ø1 ØF CD 21 A1 4D 1Ø 2B Ø6 4F 12 7D Ø2 F7 D8 3D
```

```
[FM5]   39: FF ØF 7F 1Ø B9 1Ø ØØ F7 E1 E8 99 ØØ B4 Ø9 BA 47
[EP5]   4Ø: ØF CD 21 C3 B4 Ø9 BA 24 ØF CD 21 C3 5Ø B4 Ø9 BA
[EL5]   41: F5 ØF CD 21 58 EB 3C 9Ø AØ B9 ØC ØA CØ 75 32 B4
[SN5]   42: 3C B9 ØØ ØØ BA BC ØC CD 21 72 E1 A3 51 14 8B D8
[RM5]   43: B4 4Ø B9 Ø2 Ø2 9Ø BA 4D 1Ø CD 21 72 CF B4 3E 8B
[DT5]   44: 1E 51 14 CD 21 72 C5 B4 Ø9 BA C8 ØF CD 21 EB Ø1
[PG5]   45: 9Ø BØ ØØ B4 4C CD 21 B3 Ø4 5Ø 53 B1 Ø4 D2 EC 8Ø
[WH5]   46: FC ØA 7C Ø3 8Ø C4 Ø7 8Ø C4 3Ø 8A D4 B4 Ø2 CD 21
[NW5]   47: 5B 58 B1 Ø4 D3 EØ FE CB 75 DF C3 8A EØ B3 Ø2 EB
[7P5]   48: D8 8B Ø4 56 E8 DØ FF B4 Ø2 B2 3A CD 21 5E 8B 44
[7M5]   49: Ø2 E8 C3 FF C3 B9 1Ø 27 B3 ØØ BA ØØ ØØ F7 F1 52
[TK5]   5Ø: 3A C3 74 ØE B3 FF 53 51 Ø4 3Ø 8A DØ B4 Ø2 CD 21
[6P5]   51: 59 5B 8B C1 B9 ØA ØØ BA ØØ ØØ F7 F1 8B C8 58 83
[4V5]   52: F9 Ø1 75 Ø4 B3 FF EB D2 ØB C9 75 CE C3 3C 77 7F
[1E5]   53: 16 B4 ØØ Ø3 CØ Ø5 58 Ø4 5Ø B4 Ø9 BA 42 ØF CD 21
[ØG5]   54: B4 Ø9 5E 8B 14 CD 21 C3 48 Ø5 62 Ø5 76 Ø5 94 Ø5
[FH5]   55: B9 Ø5 DA Ø5 FF Ø5 31 Ø6 52 Ø6 77 Ø6 97 Ø6 C6 Ø6
[XF5]   56: E3 Ø6 ØØ Ø7 1D Ø7 3B Ø7 58 Ø7 78 Ø7 9B Ø7 BB Ø7
[EK5]   57: FØ Ø7 2Ø Ø8 55 Ø8 6E Ø8 86 Ø8 AA Ø8 BC Ø8 CF Ø8
[8K5]   58: EB Ø8 Ø3 Ø9 2A Ø9 4F Ø9 78 Ø9 9Ø Ø9 A7 Ø9 D1 Ø9
[ST5]   59: F8 Ø9 1C ØA 3B ØA 5E ØA 8F ØA 8F ØA 8F ØA 8F ØA
[SW5]   6Ø: 8F ØA 8F ØA 8F ØA 8F ØA 8F ØA 8F ØA 8F ØA 8F ØA
[SW5]   61: 8F ØA 8F ØA 8F ØA 8F ØA 8F ØA 8F ØA 8F ØA 8F ØA
[TU5]   62: 8F ØA 8F ØA 8F ØA 8F ØA C4 ØA E1 ØA ØA ØB ØA ØB
[SK5]   63: ØA ØB ØA ØB 17 ØB 41 ØB 41 ØB 41 ØB 58 ØB 72 ØB
[SK5]   64: 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB
[SK5]   65: 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB
[SK5]   66: 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB 72 ØB
[WM5]   67: 72 ØB 72 ØB 72 ØB 72 ØB 89 ØB 89 ØB 89 ØB 89 ØB
[1S5]   68: 89 ØB 89 ØB 89 ØB 89 ØB 95 ØB B4 ØB E9 ØB ØC ØC
[1M5]   69: 2F ØC 4E ØC 6E ØC 99 ØC 48 57 3A 2Ø 44 69 76 69
[AK5]   7Ø: 73 69 6F 6E 2Ø 64 75 72 63 68 2Ø 4E 75 6C 6C ØD
[NH5]   71: ØA 24 48 57 3A 2Ø 45 69 6E 7A 65 6C 73 63 68 72
[GK5]   72: 69 74 74 ØD ØA 24 48 57 3A 2Ø 4E 4D 49 2Ø 28 7A
[TB5]   73: 2E 42 2E 2Ø 5Ø 61 72 69 74 79 2Ø 45 72 72 6F 72
[5D5]   74: 29 ØD ØA 24 48 57 3A 2Ø 42 65 68 61 6E 64 6C 75
[AF5]   75: 6E 67 2Ø 31 2D 42 79 74 65 2Ø 53 6F 66 74 77 61
[RF5]   76: 72 65 2D 49 4E 54 ØD ØA 24 48 57 3A 2Ø 9A 62 65
[7D5]   77: 72 6C 61 75 66 2Ø 2F 2Ø 42 65 68 61 6E 64 6C 75
[DH5]   78: 6E 67 2Ø 49 4E 54 4F ØD ØA 24 48 57 3A 2Ø 5Ø 72
[XC5]   79: 69 6E 74 2Ø 53 63 72 65 65 6E 2Ø 2F 2Ø 61 62 2Ø
[ØE5]   8Ø: 38 3Ø 31 38 36 3A 2Ø 42 4F 55 4E 44 ØD ØA 24 48
[2D5]   81: 57 3A 2Ø 4D 61 73 6B 69 65 72 62 61 72 65 72 2Ø
[KC5]   82: 49 6E 74 65 72 72 75 7Ø 74 2Ø 2F 2Ø 55 6E 67 81
[AK5]   83: 6C 74 69 67 65 72 2Ø 4F 7Ø 2D 43 6F 64 65 ØD ØA
[FC5]   84: 24 48 57 3A 2Ø 61 62 2Ø 38 3Ø 32 38 36 3A 2Ø 6B
[EH5]   85: 65 69 6E 2Ø 43 6F 7Ø 72 6F 7A 65 73 73 6F 72 ØD
[3C5]   86: ØA 24 48 57 3A 2Ø 48 61 72 64 77 61 72 65 2D 54
[4E5]   87: 69 6D 65 72 2D 49 6E 74 65 72 72 75 7Ø 74 3A 2Ø
[XB5]   88: 49 52 51 3Ø ØD ØA 24 48 57 3A 2Ø 54 61 73 74 61
[4F5]   89: 74 75 72 2D 49 6E 74 65 72 72 75 7Ø 74 3A 2Ø 49
[EA5]   9Ø: 52 51 2Ø 31 ØD ØA 24 48 57 3A 2Ø 49 52 51 2Ø 32
[6D5]   91: 3A 2Ø 32 74 65 72 2Ø 49 6E 74 65 72 72 75 7Ø 74
[DM5]   92: 2D 43 6F 6E 74 72 6F 6C 6C 65 72 2Ø 62 65 69 6D
[VG5]   93: 2Ø 41 54 ØD ØA 24 48 57 3A 2Ø 43 4F 4D 32 2D 49
[8B5]   94: 6E 74 65 72 72 75 7Ø 74 2Ø 28 49 52 51 2Ø 33 29
[CK5]   95: ØD ØA 24 48 57 3A 2Ø 43 4F 4D 31 2D 49 6E 74 65
[R95]   96: 72 72 75 7Ø 74 2Ø 28 49 52 51 2Ø 34 29 ØD ØA 24
[UD5]   97: 48 57 3A 2Ø 4C 5Ø 54 32 2D 49 6E 74 65 72 72 75
[ZB5]   98: 7Ø 74 2Ø 28 49 52 51 2Ø 35 29 ØD ØA 24 48 57 3A
[8N5]   99: 2Ø 44 69 73 6B 2D 43 6F 6E 74 72 6F 6C 6C 65 72
[BD5]  1ØØ: 2Ø 28 49 52 51 2Ø 36 29 ØD ØA 24 48 57 3A 2Ø 4C
[4D5]  1Ø1: 5Ø 54 31 2D 49 6E 74 65 72 72 75 7Ø 74 2Ø 28 49
[UB5]  1Ø2: 52 51 2Ø 37 29 ØD ØA 24 42 49 4F 53 3A 2Ø 42 69
[4P5]  1Ø3: 6C 64 73 63 68 69 72 6D 2D 4D 61 6E 69 7Ø 75 6C
[7K5]  1Ø4: 61 74 69 6F 6E ØD ØA 24 42 49 4F 53 3A 2Ø 56 6F
[ND5]  1Ø5: 72 68 61 6E 64 65 6E 65 2Ø 48 61 72 64 77 61 72
[UK5]  1Ø6: 65 2Ø 6D 65 6C 64 65 6E ØD ØA 24 42 49 4F 53 3A
[6C5]  1Ø7: 2Ø 53 7Ø 65 69 63 68 65 72 67 72 94 E1 65 2Ø 65
[SK5]  1Ø8: 72 6D 69 74 74 65 6C 6E ØD ØA 24 42 49 4F 53 3A
[BG5]  1Ø9: 2Ø 42 65 68 61 6E 64 6C 75 6E 67 2Ø 44 69 73 6B
[ZE5]  11Ø: 65 74 74 65 6E 2D 2Ø 2F 2Ø 46 65 73 74 7Ø 6C 61
[PH5]  111: 74 74 65 6E 2D 4C 61 75 66 77 65 72 6B ØD ØA 24
[4F5]  112: 42 49 4F 53 3A 2Ø 42 65 68 61 6E 64 6C 75 6E 67
[5D5]  113: 2Ø 64 65 72 2Ø 73 65 72 69 65 6C 6C 65 6E 2Ø 53
[DG5]  114: 63 68 6E 69 74 74 73 74 65 6C 6C 65 6E ØD ØA 24
[SF5]  115: 42 49 4F 53 3A 2Ø 53 7Ø 65 63 69 61 6C 2D 49 6E
[UD5]  116: 74 65 72 65 73 74 2D 49 6E 74 65 72 72 75 7Ø 74
[GC5]  117: 2Ø 28 6A 65 2Ø 6E 61 63 68 2Ø 52 65 63 68 6E 65
[6D5]  118: 72 29 ØD ØA 24 42 49 4F 53 3A 2Ø 54 61 73 74 61
[YE5]  119: 74 75 72 2D 41 62 66 72 61 67 65 ØD ØA 24 42 49
[5F5]  12Ø: 4F 53 3A 2Ø 44 72 75 63 6B 65 72 2D 41 75 73 67
[FF5]  121: 61 62 65 ØD ØA 24 52 4F 4D 2D 42 41 53 49 43 2Ø
[FE5]  122: 28 6E 75 72 2Ø 62 65 69 2Ø 4F 72 69 67 69 6E 61
[5H5]  123: 6C 2D 49 42 4D 73 29 ØD ØA 24 42 49 4F 53 3A 2Ø
[FC5]  124: 42 6F 6F 74 73 74 72 61 7Ø ØD ØA 24 42 49 4F 53
[5H5]  125: 3A 2Ø 53 79 73 74 65 6D 7A 65 69 74 ØD ØA 24 42
[TH5]  126: 49 4F 53 2F 44 4F 53 3A 2Ø 43 74 72 6C 2D 42 72
[XK5]  127: 6B 2D 56 65 6B 74 6F 72 ØD ØA 24 42 49 4F 53 3A
[ØF5]  128: 2Ø 54 69 6D 65 72 2D 49 6E 74 65 72 72 75 7Ø 74
[KG5]  129: ØD ØA 24 42 49 4F 53 3A 2Ø 42 69 6C 64 73 63 68
[ED5]  13Ø: 69 72 6D 2D 5Ø 61 72 61 6D 65 74 65 72 2Ø 28 41
[RE5]  131: 64 72 65 73 73 65 29 ØD ØA 24 42 49 4F 53 2F 44
[RF5]  132: 4F 53 3A 2Ø 44 69 73 6B 2D 5Ø 61 72 61 6D 65 74
```

```
[UB5]   133: 65 72 2Ø 28 41 64 72 65 73 73 65 29 ØD ØA 24 42
[GK5]   134: 49 4F 53 2F 44 4F 53 3A 2Ø 47 72 61 66 69 6B 2D
[UC5]   135: 5A 65 69 63 68 65 6E 73 61 74 7A 2Ø 28 41 64 72
[MF5]   136: 65 73 73 65 29 ØD ØA 24 44 4F 53 3A 2Ø 5Ø 72 6F
[SH5]   137: 67 72 61 6D 6D 2Ø 62 65 65 6E 64 65 6E ØD ØA 24
[LL5]   138: 44 4F 53 2D 46 75 6E 6B 74 69 6F 6E 73 61 75 66
[SD5]   139: 72 75 66 65 ØD ØA 24 44 4F 53 3A 2Ø 5Ø 72 6F 67
[7L5]   14Ø: 72 61 6D 6D 65 6E 64 65 2D 42 65 68 61 6E 64 6C
[HG5]   141: 75 6E 67 2Ø 28 5Ø 72 6F 7A 65 64 75 72 29 ØD ØA
[FN5]   142: 24 44 4F 53 3A 2Ø 43 6F 6E 74 72 6F 6C 2D 43 2D
[AG5]   143: 42 65 68 61 6E 64 6C 75 6E 67 2Ø 28 5Ø 72 6F 7A
[ØE5]   144: 65 64 75 72 29 ØD ØA 24 44 4F 53 3A 2Ø 46 65 68
[RH5]   145: 6C 65 72 2D 42 65 68 61 6E 64 6C 75 6E 67 2Ø 28
[TF5]   146: 5Ø 72 6F 7A 65 64 75 72 29 ØD ØA 24 44 4F 53 3A
[YB5]   147: 2Ø 44 69 73 6B 65 74 74 65 2Ø 2F 2Ø 5Ø 6C 61 74
[TG5]   148: 74 65 2Ø 6C 65 73 65 6E ØD ØA 24 44 4F 53 3A 2Ø
[GE5]   149: 44 69 73 6B 65 74 74 65 2Ø 2F 2Ø 5Ø 6C 61 74 74
[GG5]   15Ø: 65 2Ø 73 63 68 72 65 69 62 65 6E ØD ØA 24 44 4F
[8G5]   151: 53 3A 2Ø 5Ø 72 6F 67 72 61 6D 6D 65 6E 64 65 2C
[CK5]   152: 2Ø 5Ø 72 6F 67 72 61 6D 6D 2Ø 62 6C 65 69 62 74
[6C5]   153: 2Ø 69 6D 2Ø 53 7Ø 65 69 63 68 65 72 ØD ØA 24 44
[ØC5]   154: 4F 53 3A 2Ø 52 65 73 65 72 76 69 65 72 74 2Ø 66
[CK5]   155: 81 72 2Ø 6E 69 63 68 74 2Ø 64 6F 6B 75 6D 65 6E
[AF5]   156: 74 69 65 72 74 65 2Ø 46 75 6E 6B 74 69 6F 6E 65
[7F5]   157: 6E ØD ØA 24 42 49 4F 53 3A 2Ø 44 69 73 6B 65 74
[8N5]   158: 74 65 6E 2D 46 75 6E 6B 74 69 6F 6E 65 6E ØD ØA
[V95]   159: 24 41 64 72 65 73 73 65 2Ø 64 65 72 2Ø 65 72 73
[GE5]   16Ø: 74 65 6E 2Ø 46 65 73 74 7Ø 6C 61 74 74 65 6E 2D
[7G5]   161: 54 61 62 65 6C 6C 65 ØD ØA 24 52 65 73 65 72 76
[6C5]   162: 69 65 72 74 ØD ØA 24 41 64 72 65 73 73 65 2Ø 64
[MD5]   163: 65 72 2Ø 7A 77 65 69 74 65 6E 2Ø 46 65 73 74 7Ø
[TH5]   164: 6C 61 74 74 65 6E 2D 54 61 62 65 6C 6C 65 ØD ØA
[PD5]   165: 24 46 72 65 69 2Ø 66 81 72 2Ø 41 6E 77 65 6E 64
[MD5]   166: 75 6E 67 65 6E ØD ØA 24 41 54 3A 2Ø 41 6C 61 72
[EG5]   167: 6D 2D 5A 65 69 74 2Ø 65 72 72 65 69 63 68 74 ØD
[EE5]   168: ØA 24 46 72 65 69 2Ø 66 81 72 2Ø 41 6E 77 65 6E
[YF5]   169: 64 75 6E 67 65 6E ØD ØA 24 55 6E 62 65 6E 75 74
[ZE5]   17Ø: 7A 74 ØD ØA 24 41 54 2D 42 49 4F 53 3A 2Ø 45 63
[DC5]   171: 68 74 7A 65 69 74 75 68 72 2Ø 28 49 52 51 2Ø 38
[2D5]   172: 29 ØD ØA 24 41 54 2D 42 49 4F 53 3A 2Ø 41 75 66
[LA5]   173: 72 75 66 2Ø 49 4E 54 3Ø 41 2C 2Ø 64 61 64 75 72
[4M5]   174: 63 68 2Ø 53 69 6D 75 6C 61 74 69 6F 6E 2Ø 76 6F
[KD5]   175: 6E 2Ø 49 52 51 32 ØD ØA 24 41 54 2D 42 49 4F 53
[5H5]   176: 3A 2Ø 43 4F 4D 33 2D 49 6E 74 65 72 72 75 7Ø 74
[185]   177: 2Ø 28 49 52 51 2Ø 31 3Ø 29 ØD ØA 24 41 54 2D 42
[7K5]   178: 49 4F 53 3A 2Ø 43 4F 4D 34 2D 49 6E 74 65 72 72
[Z95]   179: 75 7Ø 74 2Ø 28 49 52 51 2Ø 31 31 29 ØD ØA 24 41
[DD5]   18Ø: 54 2D 42 49 4F 53 3A 2Ø 52 65 73 65 72 76 69 65
[PA5]   181: 72 74 2Ø 28 49 52 51 2Ø 31 32 29 ØD ØA 24 41 54
[BL5]   182: 2D 42 49 4F 53 3A 2Ø 43 6F 7Ø 72 6F 7A 65 73 73
[EC5]   183: 6F 72 2Ø 28 49 52 51 2Ø 31 33 29 ØD ØA 24 41 54
[SF5]   184: 2D 42 49 4F 53 3A 2Ø 46 65 73 74 7Ø 6C 61 74 74
[VH5]   185: 65 6E 2D 43 6F 6E 74 72 6F 6C 6C 65 72 2Ø 28 49
[LB5]   186: 52 51 2Ø 31 34 29 ØD ØA 24 41 54 2D 42 49 4F 53
[9C5]   187: 3A 2Ø 52 65 73 65 72 76 69 65 72 74 2Ø 28 49 52
[B85]   188: 51 2Ø 31 35 29 ØD ØA 24 ØØ ØØ ØØ ØØ 49 4E 54 53
[ZE5]   189: 2E 41 44 52 ØØ 46 65 68 6C 65 72 2Ø 69 6E 2Ø 4B
[7M5]   19Ø: 6F 6D 6D 61 6E 64 6F 7A 65 69 6C 65 ØD ØA ØA 24
[BE5]   191: 5Ø 72 6F 67 72 61 6D 6D 2Ø 7A 75 72 2Ø 55 6E 74
[UF5]   192: 65 72 73 75 63 68 75 6E 67 2Ø 64 65 72 2Ø 49 6E
[PH5]   193: 74 65 72 72 75 7Ø 74 2D 56 65 6B 74 6F 72 65 6E
[5H5]   194: ØD ØA ØA 53 79 6E 74 61 78 3A 2Ø 43 48 4B 49 4E
[7D5]   195: 54 53 2Ø 2Ø 2Ø 2Ø 2Ø 48 69 6C 66 65 3A 2Ø 4B 6F
[WD5]   196: 72 72 65 6B 74 65 2Ø 53 79 6E 74 61 78 2Ø 61 75
[D85]   197: 73 67 65 62 65 6E ØD ØA 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
[8E5]   198: 43 48 4B 49 4E 54 53 2Ø 2F 4C 2Ø 2Ø 61 6B 74 75
[2F5]   199: 65 6C 6C 65 2Ø 49 6E 74 65 72 72 75 7Ø 74 2D 4C
[ED5]   2ØØ: 69 73 74 65 2Ø 61 75 66 2Ø 42 69 6C 64 73 63 68
[5E5]   2Ø1: 69 72 6D 2Ø 61 75 73 67 65 62 65 6E ØD ØA 2Ø 2Ø
[F95]   2Ø2: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 43 48 4B 49 4E 54 53 2Ø 2F 57
[ØH5]   2Ø3: 2Ø 2Ø 61 6B 74 75 65 6C 6C 65 2Ø 49 6E 74 65 72
[HE5]   2Ø4: 72 75 7Ø 74 2D 4C 69 73 74 65 2Ø 69 6E 2Ø 44 61
[EC5]   2Ø5: 74 65 69 2Ø 73 63 68 72 65 69 62 65 6E ØD ØA 2Ø
[2B5]   2Ø6: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 43 48 4B 49 4E 54 53 2Ø 2F
[UE5]   2Ø7: 43 2Ø 2Ø 61 6B 74 75 65 6C 6C 65 2Ø 4C 69 73 74
[KB5]   2Ø8: 65 2Ø 6D 69 74 2Ø 44 61 74 65 69 2Ø 76 65 72 67
[TE5]   2Ø9: 6C 65 69 63 68 65 6E 2Ø 75 6E 64 2Ø 55 6E 74 65
[K55]   21Ø: 72 2D ØD ØA 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
[D65]   211: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 73 63 68 69 65 64 65 2Ø
[GB5]   212: 61 75 66 2Ø 42 69 6C 64 73 63 68 69 72 6D 2Ø 61
[6E5]   213: 75 73 67 65 62 65 6E ØD ØA ØA 2Ø 2Ø 2Ø 2Ø 41 6C
[NC5]   214: 6C 65 2Ø 5Ø 61 72 61 6D 65 74 65 72 2Ø 64 81 72
[6E5]   215: 66 65 6E 2Ø 61 75 63 68 2Ø 67 65 6D 69 73 63 68
[VE5]   216: 74 2Ø 77 65 72 64 65 6E ØD ØA ØA 24 2Ø 49 6E 74
[8F5]   217: 65 72 72 75 7Ø 74 2D 56 65 6B 74 6F 72 2D 54 61
[FR5]   218: 62 65 6C 6C 65 3A ØD ØA C4 C4 C4 C4 C4 C4 C4 C4
[SV5]   219: C4 C4 C4 C4 C4 C4 C4 C4 C4 C4 C4 C4 C4 C4 C4 C4
[DN5]   22Ø: C4 C4 C4 ØD ØA ØA 41 6B 74 75 65 6C 6C 65 73 2Ø
[KF5]   221: 43 6F 64 65 2D 53 65 67 6D 65 6E 74 3A 2Ø 24 2Ø
[LE5]   222: 41 6C 74 65 73 2Ø 43 6F 64 65 2D 53 65 67 6D 65
[YD5]   223: 6E 74 3A 2Ø 24 68 2C 2Ø 61 6B 74 75 65 6C 6C 65
[LE5]   224: 73 2Ø 43 6F 64 65 2D 53 65 67 6D 65 6E 74 3A 2Ø
[FB5]   225: 24 68 ØD ØA 2Ø 2Ø 2Ø 2Ø 55 6E 74 65 72 73 63 68
[VC5]   226: 69 65 64 2Ø 69 6E 2Ø 42 79 74 65 73 3A 2Ø 24 68
```

```
[VB5]   227: ØD ØA ØA 24 6D 65 68 72 2Ø 61 6C 73 2Ø 36 34 2Ø
[8E5]   228: 4B ØD ØA 24 41 64 72 28 49 6E 74 2Ø 24 29 2Ø 3D
[3C5]   229: 2Ø 24 2Ø 2Ø 2Ø 2Ø 24 ØD ØA 24 55 6E 74 65 72 73
[UD5]   23Ø: 63 68 69 65 64 65 2Ø 7A 77 69 73 63 68 65 6E 2Ø
[WD5]   231: 61 6C 74 65 72 2Ø 75 6E 64 2Ø 6E 65 75 65 72 2Ø
[EF5]   232: 49 6E 74 65 72 72 75 7Ø 74 2D 56 65 6B 74 6F 72
[CF5]   233: 2D 54 61 62 65 6C 6C 65 3A ØD ØA 24 2Ø 49 6E 74
[G95]   234: 65 72 72 75 7Ø 74 3A 2Ø 24 68 3A 2Ø 2Ø 41 6C 74
[DF5]   235: 65 72 2Ø 56 65 6B 74 6F 72 3A 2Ø 24 2C 2Ø 6E 65
[2C5]   236: 75 65 72 2Ø 56 65 6B 74 6F 72 3A 2Ø 24 2Ø 4B 65
[FE5]   237: 69 6E 65 21 ØD ØA ØA 24 4E 65 75 65 2Ø 44 61 74
[HF5]   238: 65 69 2Ø 49 4E 54 53 2E 41 44 52 2Ø 65 72 66 6F
[RE5]   239: 6C 67 72 65 69 63 68 2Ø 65 72 73 74 65 6C 6C 74
[HF5]   24Ø: 21 ØD ØA ØA 24 46 65 68 6C 65 72 2Ø 62 65 69 6D
[8H5]   241: 2Ø 44 61 74 65 69 2D 48 61 6E 64 6C 69 6E 67 ØD
[SC5]   242: ØA ØA 24 44 61 74 65 69 2Ø 49 4E 54 53 2E 41 44
[ØG5]   243: 52 2Ø 69 6D 2Ø 61 6B 74 75 65 6C 6C 65 6E 2Ø 56
[CH5]   244: 65 72 7A 65 69 63 68 6E 69 73 2Ø 6E 69 63 68 74
[6D5]   245: 2Ø 67 65 66 75 6E 64 65 6E ØD ØA ØA 24 ØØ ØØ ØØ
[ØZ4]   246: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   247: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   248: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   249: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   25Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   251: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   252: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   253: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   254: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   255: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   256: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   257: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   258: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   259: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   26Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   261: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   262: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   263: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   264: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   265: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   266: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   267: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   268: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   269: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   27Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   271: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   272: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   273: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   274: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   275: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   276: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   277: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   278: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   279: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   28Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   281: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   282: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   283: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   284: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   285: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   286: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   287: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   288: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   289: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   29Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   291: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   292: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   293: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   294: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   295: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   296: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   297: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   298: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   299: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3ØØ: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø1: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø2: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø3: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø4: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø5: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø6: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø7: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø9: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   31Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   311: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   312: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   313: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   314: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   315: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØR3]   316: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
```

Listing 2. Die Hexdump-Version von »chkints.com« zur Eingabe mit SCheck

Bringen Sie Tempo in die Bildschirmausgabe

Als Assembler-Programmierer kennen Sie das Problem: Die Bildschirmausgabe über die üblichen MS-DOS-Interrupts ist furchtbar langsam. Mit einer kurzen Unterroutine, die direkt in den Bildschirmspeicher schreibt, beschleunigen Sie Ihre Programme problemlos um den Faktor zehn und mehr.

Um eine längere Zeichenkette auf dem Bildschirm auszugeben, benötigt ein PC mit einer Taktfrequenz von 4,77 MHz schon einige Sekunden; zu viel, um bildschirmintensive Anwendungen (wie zum Beispiel eine Textverarbeitung) zu programmieren. Wenn Sie in Assembler programmieren, kennen Sie sicher die Funktion 09hex des MS-DOS-Interrupt 21hex. Mit diesem Aufruf geben Sie sehr einfach Zeichenketten auf dem Bildschirm aus. Dazu müssen Sie nur den Offset des auszugebenden Texts in das Register DX laden und dann diese MS-DOS-Funktion aufrufen:

```
mov ah,09h
mov dx,offset meldung
int 21h
meldung: db 'DOS International$'
```

Das Dollarzeichen muß das Ende der Zeichenkette kennzeichnen. Allerdings ist diese Funktion – wie bereits erwähnt – sehr langsam, und schon bei der Ausgabe eines 200 Zeichen langen Texts fällt die niedrige Geschwindigkeit auf. Wenn Sie gar den ganzen Bildschirm (80x25 = 2000 Zeichen) beschreiben wollen, müssen Sie auf einem Standard-PC fast drei Sekunden warten.

Die extrem langsame Ausführung ist unter anderem darauf zurückzuführen, daß die Funktion 09hex bei der Ausführung andere MS-DOS-Interrupts verwendet. Es gibt aber eine recht simple Alternative, um eine sehr schnelle Bildschirmausgabe zu erreichen – nämlich das direkte Schreiben der ASCII-Zeichen in den Bildschirmspeicher mit Hilfe des Assembler-Befehls »mov«.

Bei einer Hercules-Grafikkarte ist das auch ohne weiteres möglich, da sich der Video-Controller und der Prozessor des PC beim Zugriff nicht in die Quere kommen. Sie müssen also nur das ES-Register mit dem Segment des Bildschirmspeichers laden (in diesem Falle B000hex), die Adresse des Zeichens berechnen und den ASCII-Wert des Zeichens, das Sie darstellen wollen, in diese Adresse schreiben. Die Adresse einer beliebigen Bildschirmposition errechnen Sie mit der Formel

```
160 x (zeile - 1) + 2 x (spalte - 1)
```

wobei »zeile« ein Wert zwischen 1 und 25 und »spalte« ein Wert zwischen 1 und 80 sein muß. Wie Sie an der Ziffer 2 in der Formel unschwer erkennen, belegt ein Zeichen immer zwei Byte. Das erste enthält den ASCII-Wert des Zeichens und das zweite Byte das Attribut zu diesem Zeichen.

Folgende Werte für das Attributbyte sind zulässig:

```
Schwarze Anzeige: 00hex
Unterstrichen:    01hex
Normale Anzeige:  07hex
Inverse Anzeige:  70hex
```

Wenn die Blinkfunktion über die Software eingeschaltet ist (Bit 5 des Ausgabeports 03B8hex ist gesetzt), erhalten Sie ein blinkendes Zeichen, indem Sie zu dem Attributwert 80hex addieren. Übertragen Sie also den ASCII-Wert des gewünschten Zeichens in die errechnete Speicherzelle und das dazugehörige Attributbyte in die nächste.

Etwas anders sieht es bei einer CGA-Grafikkarte aus. Wenn Sie beispielsweise in Basic mit einigen Poke-Befehlen in den Bildschirmspeicher schreiben, ist die Bildschirmausgabe mit »Schneetreiben« verbunden; das heißt, während des Zugriffs erscheinen undefinierte Punkte auf dem Bildschirm.

Dieser negative Effekt ist darauf zurückzuführen, daß sowohl der CRT-(Video)-Controller der Grafikkarte als auch der Prozessor des Computers auf den Bildschirmspeicher zugreifen, und dabei Kollisionen auftreten, die sich dann als »Schnee« bemerkbar machen.

Um diese Interferenzen zu vermeiden, gibt es zwei Verfahren:

- Während das Programm auf den Bildschirmspeicher zugreift, schalten Sie die Bildschirmanzeige über die Software kurz ab; die »verschneite« Anzeige wird also unterdrückt.
- Warten Sie, bevor Sie auf den Bildschirmspeicher zugreifen, bis der Strahlrücklauf (»vertical retrace«) aktiv ist. Als Strahlrücklauf bezeichnet man den Zeitpunkt, an dem der Elektronenstrahl des Monitors vom Ende der gerade gezeichneten Bildschirmzeile zum Anfang der nächsten läuft. Damit dieser Vorgang keine störenden Linien auf den Bildschirm bringt, wird die Elektronenkanone dabei abgedunkelt (der Elektronenstrahl also unterdrückt).

Das erste Verfahren hat den Nachteil, daß ein störendes Flimmern auftritt, wenn der Bildschirm mit Zeichen gefüllt oder hardwarebedingt invertiert ist (zum Beispiel bei einem Weißschwarz-Monitor), weil Sie das kurzzeitige Abdunkeln einer größeren hellen Fläche sehr stark wahrnehmen. Der Vorteil gegenüber der zweiten Methode besteht darin, daß keine unnötige Rechenzeit mit dem Warten auf den Strahlrücklauf vergeudet wird.

Der direkte Zugriff auf den Bildschirmspeicher beschleunigt Ihre Programme enorm

In diesem Zusammenhang soll das zweite Verfahren näher erläutert werden. Auch hierbei ist der Geschwindigkeitszuwachs erstaunlich groß.

Wenn Sie direkt auf den Bildschirmspeicher der CGA-Grafikkarte zugreifen möchten, müssen Sie folgendermaßen vorgehen:

- Laden Sie das AL-Register mit dem ASCII-Wert des auszugebenden Zeichens und das AH-Register mit dem Attribut für dieses Zeichen.
- Berechnen Sie die Adresse des Zeichens aus den Koordinaten.

Farbe	Wert
Schwarz	0
Blau	1
Grün	2
Kobaltblau	3
Rot	4
Violett	5
Braun	6
Weiß	7
Grau	8
Hellblau	9
Hellgrün	10
Hellkobaltblau	11
Hellrot	12
Hellviolett	13
Gelb	14
intensives Weiß	15

Tabelle. Die CGA-Karte erlaubt 16 verschiedene Farben für den Vorder- und Hintergrund

- Warten Sie, bis der Strahlrücklauf aktiv ist.
- Schreiben Sie nun das Zeichen in die errechnete Adresse und sein Attribut in die nächste. Dieser Vorgang ist sehr zeitkritisch, weil er nicht länger als der Strahlrücklauf dauern darf.

Die Assembler-Routine übernimmt all diese Aufgaben und verhält sich dem Programmierer gegenüber fast wie die Funktion 09hex des MS-DOS-Interrupt 21hex; das heißt, Sie müssen beim Aufruf folgende Parameter übergeben:

```
DS:BX – Startadresse der auszugebenden Zeichenkette
DL – Spalte (im Bereich 1 bis 80)
DH – Zeile (im Bereich 1 bis 25)
```

Das letzte Zeichen der Zeichenkette muß ein Dollarzeichen »$« sein.

Die Adresse einer beliebigen Bildschirmposition errechnen Sie im 80x25-Textmodus der CGA-Grafikkarte mit der gleichen Formel wie bei der Hercules-Karte, also mit der Formel

```
160 x (zeile - 1) + 2 x (spalte - 1)
```

Im Gegensatz zur Hercules-Karte beginnt der Bildschirmspeicher der CGA-Karte allerdings bei der Segmentadresse B800hex (Hercules-Karte: B000hex). Auch bei der CGA-Karte stehen für jede Bildschirmposition zwei Byte zur Verfügung. Das erste enthält den ASCII-Wert, das zweite das Attribut für diese Position. Die zulässigen (und sinnvollen) Werte für das Attribut unterscheiden sich aber von den Attributen für die Hercules-Karte. Den Wert für das jeweilige Attributbyte errechnen Sie nach der Formel

```
16 x Hintergrundfarbe + Schriftfarbe
```

wobei »Schriftfarbe« und »Hintergrundfarbe« Werte zwischen 0 und 15 annehmen dürfen. Die Werte der 16 verschiedenen Farben entnehmen Sie der Tabelle. Die Hintergrundfarben zwischen 8 und 15 haben zwei unterschiedliche Funktionen: Entweder stehen sie für die helle Ausführung der Farben 0 bis 7 oder sie entsprechen den Farben 0 bis 7 und lassen zusätzlich das dazugehörige Zeichen blinken. Die beiden Funktionen lassen sich über den Ausgabeport 03D8hex (»mode register«) umschalten. Wenn Bit 5 dieses Ports gesetzt ist, ist die Blinkfunktion aktiviert; ist es gelöscht, repräsentieren die Werte zwischen 8 und 15 die intensivierten Farben 0 bis 7. Die Routine »quickscr.asm« setzt die Schriftfarbe immer auf Weiß und den Hintergrund auf Schwarz. Wenn Sie eine eigene Farbenkombination verwenden möchten, ändern Sie einfach den Wert 0Fhex, den die Routine zweimal verwendet.

Um festzustellen, ob der Strahlrücklauf gerade aktiv ist, müssen Sie das »display status register« (Adresse 03DAhex) der CGA-Grafikkarte lesen. Das Bit 0 gibt an, ob der Strahlrücklauf derzeit aktiv ist (Bit 0 ist gesetzt) oder nicht (Bit 0 ist gelöscht). Folglich muß das Programm ein Unterprogramm enthalten, das wartet, bis dieses Bit gesetzt ist. Abschließend schreibt die Routine den Inhalt des Registers AX (ASCII-Wert und Attributbyte) in die durch das DI-Register adressierte Speicherzelle (Befehl »stosw«).

Das Beispielprogramm füllt den ganzen Bildschirm unter Verwendung der Routine »qscrn« mit Herz-Zeichen und gibt zwischendurch einen kurzen Text aus. An diesem Beispiel sehen Sie sehr deutlich den Geschwindigkeitszuwachs im Vergleich zu der MS-DOS-Funktion: MS-DOS braucht für ein ähnliches Programm drei Sekunden, diese Routine nur den Bruchteil einer Sekunde.

Wenn Sie diese Routine in eigene Programme einbinden, bedenken Sie bitte, daß

- »Quickscr« immer die erste Bildschirmseite beschreibt, unabhängig von der derzeit aktiven Bildschirmseite;
- die CGA-Grafikkarte sich immer im 80x25-Textmodus (farbig oder monochrom) befinden muß, wenn Quickscr aufgerufen wird. Das Beispielprogramm setzt vorher diesen Modus.

Fraktal Generator 3D

Neu: Die Dritte Dimension

Meisterstücke der Computergrafik

High-Speed
Höchsteffiziente Programmierung
Nur noch 7 Sekunden für das "Apfelmännchen" auf dem Amiga!

Mandelbrot- und Juliamenge
Glättungsfunktion

Super-Parallel-Projektion
Frei wählbarer horizontaler Blickwinkel mit 360 Grad:
Betrachten Sie das "Fraktalobjekt" von allen Seiten

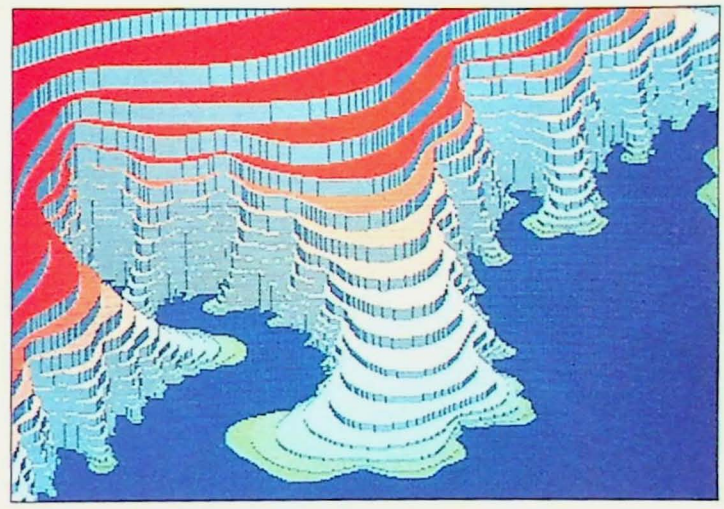
Stufenloser vertikaler Blickwinkel:
Wahlweise Sicht von oben und unten, schräg oder in der Totalen

Voller Bedienungskomfort
Auswahl komplett mit Pulldown-Menüs
Wahlweise Steuerung mit Maus oder Tastatur

Mehrere separate Bildspeicher
Der Amiga speichert im IFF-Format ab. Die Bilder lassen sich dadurch in andere Malprogramme laden.

Phantastische Farbmöglichkeiten
Separate Farbzuordnung für die einzelnen Bilder
Animationsmöglichkeit durch Color-Cycling

Die Farben lassen sich auch nachträglich beliebig verändern.
Gilt in beschränktem Maße für den Atari



PC Fraktal Generator 3D — MS-DOS ab 2.0; PC-XT/AT mit EGA-Karte oder: Amstrad/Schneider PC 1512.
MS-DOS 5 1/4"-Disk. Best.-Nr. 248
MS-DOS 3 1/2"-Disk. Best.-Nr. 249
69,– DM*

Amiga Fraktal Generator 3D — Commodore Amiga mit 512 KB
Amiga 3 1/2"-Disk. Best.-Nr. 2901
69,– DM*

Atari Fraktal Generator 3D — Atari ST
Atari 3 1/2"-Disk. Best.-Nr. 2951
69,– DM*

CPC Fraktal Generator 3D — CPC 464 / 664 / 6128
Kassette Best.-Nr. 210
59,– DM*
3"-Disk. Best.-Nr. 211
69,– DM*

Demodiskette
Best.-Nr. 001 Fraktal Generator 3 D mit MaskEdit 5,– DM

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 4,– DM bzw. 6,– DM Porto/Verpackung.
– Unverbindliche Preisempfehlung –
Bitte Bestellkarte benutzen

DMV-Verlag · Postfach 250 · 3440 Eschwege

Ein Schritt weiter als C
Zwei Schritte weiter als Pascal

LAUER & WALLWITZ
Software par excellence!
L&W

TopSpeed Modula-2

deutsche Version

Highlights des Compilers

- rekonfigurierbare Oberfläche
- Multi-Window-Editor
- schneller One-Pass-Compiler
- alle Speichermodelle mit voller Segmentkontrolle
- Intel/Microsoft-Objektfile Format
- 80x87 Inline-Code oder Emulation
- Smart-Linker und Make-Utility
- Library-Quellcode
- separate Compilierung von Modulen
- long und far Datentypen
- long/short Zeiger
- Grafikunterstützung CGA, HGC, EGA und VGA
- Extrem kompakter und schneller Code
- Scheduler für parallel laufende Prozesse

Machen auch Sie den Schritt in die Zukunft.

ISAM & Display Manager

Der im Lieferumfang des Compilers enthaltene ISAM-Manager (B-Tree) kann bis zu 4,3 Milliarden Schlüssel, Recordgrößen bis 64 KByte, Wildcards und Umlaute verarbeiten. Der gleichfalls mitgelieferte Display Manager besteht aus einem Editor zur Erstellung und Modifikation beliebiger Masken sowie einer Reihe Modula-2-Funktionen zur Nutzung der erstellten Masken in eigenen Programmen.

TechKit

Das TopSpeed Modula-2 TechKit enthält Routinen zur maschinennahen Programmierung auf IBM-kompatiblen PCs. Dies beinhaltet einen Assembler, der in die Compileroberfläche integriert werden kann, Tools zur Programmierung speicherresidenter Programme (TSR), Routinen zur Ansteuerung der seriellen Schnittstelle, einen Overlay-Manager, Funktionen zum EMS-Handling, BIOS-Interface und Locator für PROM-Programme sowie weitere Modula-2-spezifische Utilities.

Minimale Systemanforderungen:
PC/XT/AT mit 384 KB RAM, zwei Diskettenlaufwerke oder Festplatte, PC-/MS-DOS ab Version 2.0

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 4,– DM bzw. für das Ausland 6,– DM Porto und Verpackung. Bitte benutzen Sie die Bestellkarte.

VID - Visual Interface Debugger

Der Source-Level-Debugger zum TopSpeed Modula-2 erleichtert die Fehlersuche mit Einzelschrittmodus, Trace-Möglichkeiten, Zugriff auf alle Variablen – auch Strukturen, Zeiger und Aufzählungstypen – in symbolischer Form. Auch Low-Level-Zugriff auf Register und Speicherbereiche ist sehr einfach.

Um Programme kontrolliert ablaufen zu lassen, können verschiedene Arten von Unterbrechungspunkten im Quellcode gesetzt werden. Sowohl Variablen als auch Modula-2-Ausdrücke können während des Programmlaufs beobachtet werden.

Protokollierung des Zugriffs auf Variablen und eine Stapelanzeige der Prozedur-Aufrufsverschachtelung sind gleichfalls möglich.

Der Debugger beinhaltet einen symbolischen Disassembler und einen "Profiler", der eine Analyse des Laufzeitverhaltens eines Programms erlaubt.

TopSpeed Compiler inkl. ISAM & Display Manager	**348,– DM***
TopSpeed TechKit	**248,– DM***
TopSpeed Debugger	**248,– DM***
Paketpreis für Compiler und Debugger zusammen	**555,– DM***
TopSpeed Compiler OS/2-Version	**698,– DM***

DMV-Verlag · Postfach 250 · 3440 Eschwege

– Quickscr zwar nur mit einer CGA-Karte arbeitet, aber nicht geprüft wird, ob eine solche im PC enthalten ist. Wenn Sie also »nur« eine Hercules-Grafikkarte installiert haben, schreibt das Programm munter in den (nicht vorhandenen) CGA-Bildschirmspeicher. Schäden verursacht das zwar nicht, es erscheint aber keine Bildschirmanzeige.

(Martin Hepp/zi)

Literatur:
IBM PC/XT Technisches Handbuch
Monochrome Graphics Printer Adapter Card
Color-Graphics Adaptor User's Guide

Name	quickscr.asm
Sprache	MASM 5.0
Grafikkarte	CGA
Besonderheiten	keine

```
[WV4]    1: ; Programmname: QUICKSCR.ASM
[WY3]    2: ; (C) 1989 Everts & Hagedorn
[8U3]    3: ;          DOS International
[5U4]    4: ;-----------------------------------------
[XB5]    5: ; Unterroutine für die schnelle
[AC3]    6: ; Bildschirmausgabe
[7B6]    7: ; Achtung, es wird nicht überprüft, ob
[3Y5]    8: ; wirklich eine CGA-Karte im System
[PR2]    9: ; vorhanden ist!
[P3Ø]   1Ø: ;
[DC3]   11: ; Autor: Martin Hepp
[5U4]   12: ;-----------------------------------------
[ØØØ]   13:
[E36]   14:          ;Standard-Einleitung für Com-Datei
[3X3]   15: assume   cs:code,ds:code
[AD2]   16: code     segment
[3M1]   17:          org     Ø1ØØh
[ØØØ]   18:
[Z93]   19: start:   jmp     anfang
[ØØØ]   2Ø:
[5U4]   21: ;-----------------------------------------
[HF7]   22: ; Es folgt die eigentliche Prozedur »qscr«
[CK5]   23: ; Erforderliche Einsprungwerte:
[P3Ø]   24: ;
[GC5]   25: ; Startadresse des Textes = DS:BX
[H64]   26: ; letztes Zeichen         = "$"
[DC3]   27: ; Spalte                  = DL
[1A3]   28: ; Zeile                   = DH
[P3Ø]   29: ;
[DV5]   3Ø: ; <Zeile> muß zwischen 1 und 25 und
[CM5]   31: ; <Spalte> zwischen 1 und 8Ø liegen!
[P3Ø]   32: ;
[G46]   33: ; Achtung! Die Werte in DL und DH werden
[GP6]   34: ; nicht auf ihre Gültigkeit geprüft.
[KK5]   35: ; Wenn sie die zulässigen Grenzen
[1T6]   36: ; überschreiten, schreibt das Programm
[K65]   37: ; irgendwo in den Speicher des PC.
[5U4]   38: ;-----------------------------------------
[ØØØ]   39:
[DP2]   4Ø: qscrn    proc near
[PP3]   41:          ;Alle Register sichern
[YN2]   42: loop1:   push    ax
[ZF1]   43:          push    bx
[1G1]   44:          push    cx
[3G1]   45:          push    dx
[BZ5]   46:          ;weiße Schrift auf schwarzem Grund
[XY1]   47:          mov     ah,ØFh
[Z65]   48:          ;Zeichen aus dem Speicher in AL
[DL1]   49:          ;kopieren
[2V2]   5Ø:          mov     al,DS:[BX]
[C62]   51:          ;Prüfe, ob ="$"
[NW1]   52:          cmp     al,24h
[B46]   53:          ;Wenn ja, zurück zum Hauptprogramm
[CM1]   54:          jz      exit
[ØK1]   55:          ;DL = DL - 1
[F91]   56:          dec     dl
[CH1]   57:          ;DH = DH - 1
[391]   58:          dec     dh
[4U1]   59:          ;AX retten
[XF1]   6Ø:          push    ax
[F94]   61:          ;Lade AL mit dem Wert 16Ødez
[R52]   62:          mov     al,ØaØh
[PN4]   63:          ;Multipliziere Zeile mit 16Ø
[9C1]   64:          mul     dh
[ØX2]   65:          ;Ergebnis sichern
[XF1]   66:          push    ax
[H22]   67:          ;Lade AL mit 2
[1Y1]   68:          mov     al,Ø2h
[TL4]   69:          ;Multipliziere Spalte mit 2
[NC1]   7Ø:          mul     dl
[ZS4]   71:          ;Lade CX mit gesichertem Wert
[VD1]   72:          pop     cx
[R75]   73:          ;Adresse des Zeichens nun in CX
[DR1]   74:          add     cx,ax
[CU5]   75:          ;Adresse im Video-RAM in Zielindex
[DL1]   76:          ;kopieren
[5U1]   77:          mov     di,cx
[146]   78:          ;Segmentadresse des Video-RAM in ES
[NA2]   79:          mov     ax,Øb8ØØh
[PØ1]   8Ø:          ;laden
[3V1]   81:          mov     es,ax
[655]   82:          ;Adresse des »Display Status
[2Y2]   83:          ;Register« in DX
[AC3]   84: wait:    mov     dx,Ø3DAh
[2H5]   85:          ;Wert des Status-Registers lesen
[6L1]   86:          in      al,dx
[CØ2]   87:          ;Bit Ø testen
[RY1]   88:          test    al,Ø1h
[CU5]   89:          ;Wenn Strahlrücklauf nicht aktiv,
[F52]   9Ø:          ;wiederholen
[BL1]   91:          jz      wait
[N46]   92:          ;ASCII-Code des Zeichens vom Stapel
[A21]   93:          ;holen
[RD1]   94:          pop     ax
[FA5]   95:          ;Zeichen ins Video-RAM schreiben
[ZYØ]   96:          stosw
[883]   97:          ;Nächstes Zeichen
[MB1]   98:          inc     bx
[Z65]   99:          ;Zeichen aus dem Speicher in AL
[DL1]  1ØØ:          ;kopieren
[2V2]  1Ø1:          mov     al,DS:[BX]
[C62]  1Ø2:          ;Prüfe, ob ="$"
[NW1]  1Ø3:          cmp     al,24h
[B46]  1Ø4:          ;Wenn ja, zurück zum Hauptprogramm
[CM1]  1Ø5:          jz      exit
[BZ5]  1Ø6:          ;weiße Schrift auf schwarzem Grund
[XY1]  1Ø7:          mov     ah,ØFh
[XP3]  1Ø8:          ;Zeichen auf den Stack
[XF1]  1Ø9:          push    ax
[Y66]  11Ø:          ;Auf Strahlrücklauf warten und dann
[4S1]  111:          jmp     wait
[FA5]  112:          ;Zeichen ins Video-RAM schreiben
[ØØØ]  113:
[KG4]  114:          ;Register vom Stapel holen
[VH2]  115: exit:    pop     dx
[VD1]  116:          pop     cx
[TD1]  117:          pop     bx
[RD1]  118:          pop     ax
[AE5]  119:          ;und zurück zum Hauptprogramm
[3PØ]  12Ø:          ret
[9V1]  121: qscrn    endp
[ØØØ]  122:
[5U4]  123: ;-----------------------------------------
[6S6]  124: ; Nun folgt ein kurzes Demoprogramm, das
[1P6]  125: ; den Bildschirm mit Herzzeichen füllt
[6L6]  126: ; und in der Bildschirmmitte einen Text
[SD1]  127: ; ausgibt
[5U4]  128: ;-----------------------------------------
[ØØØ]  129:
[RL5]  13Ø:          ;Funktion »Bildschirmmodus auf
[YH5]  131:          ;8Øx25 farbig setzen« auswählen
[ØØØ]  132:
[BF3]  133: anfang:  mov     ax,ØØØ3h
[ØØØ]  134:
[YB5]  135:          ;und Video-Interrupt auslösen
[YH1]  136:          int     1Øh
[2C5]  137:          ;Startadresse der Meldung in BX
[2G3]  138:          mov     bx,offset text
[V42]  139:          ;erste Spalte
[5Y1]  14Ø:          mov     dl,Ø1h
[BØ2]  141:          ;erste Zeile
[TX1]  142:          mov     dh,Ø1h
[FX4]  143:          ;und nun die Meldung ausgeben
[UW1]  144:          call    qscrn
[6W5]  145:          ;MS-DOS-Funktion »End Process«
[SØ2]  146:          ;auswählen
[A62]  147:          mov     ah,Ø4ch
[A92]  148:          ;und aufrufen
[3K1]  149:          int     21h
[ØK3]  15Ø: text:    db      972 dup (3)
[C84]  151:          db      ' Quickscreen-Demo by'
[643]  152:          db      ' Martin Hepp   '
[W94]  153:          db      ' (DOS International) '
[5D2]  154:          db      973 dup (3)
[441]  155:          db      '$'
[ØØØ]  156:
[ZP1]  157: code     ends
[4W1]  158:          end     start
```

Listing. Mit der Routine »quickscr.asm« beschleunigen Sie die Bildschirmausgabe enorm

Bild 1. Mit »Easydos« haben Sie die wichtigsten MS-DOS-Funktionen direkt im Zugriff

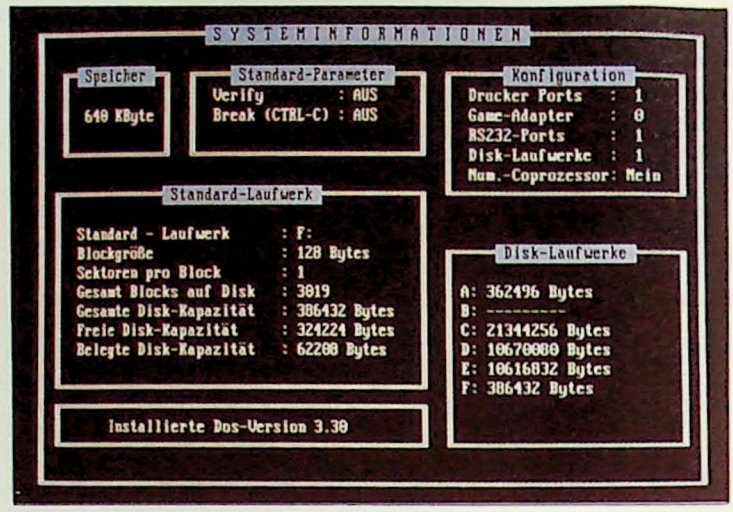


Bild 2. Die Funktion »Systeminformation« zeigt Ihnen, was in Ihrem Computer steckt

Easydos: Damit Sie es leichter haben

Neben all den Dingen, mit denen Sie sich täglich auseinandersetzen müssen, haben Sie kein Interesse daran, sich auch noch die Namen der MS-DOS-Hilfsprogramme und die Anzahl und Reihenfolge der benötigten Parameter zu merken. Außerdem sagt Ihnen ein Programm mit ansprechender Benutzerführung mehr zu als eine spartanische Kommandozeilenversion. Ganz klar: Die menügesteuerte MS-DOS-Benutzeroberfläche »Easydos« ist genau das richtige für Sie.

Easydos ist eine komplett menügesteuerte MS-DOS-Benutzeroberfläche, die alle wichtigen MS-DOS-Utilities bereits integriert hat und darüber hinaus eine Vielzahl zusätzlicher Hilfsfunktionen anbietet (Tabelle). Alle Funktionen werden dabei über die zehn Funktionstasten – zum Teil kombiniert mit [Ctrl] oder [Shift] – ausgewählt. Sofern nicht anders angegeben, werden bei Pfadabfragen Dateinamen wahlweise mit Laufwerks- und/oder Pfadangabe akzeptiert. Aus Platzgründen wurden einige Utilities nicht direkt implementiert, sondern als menügesteuerter Aufruf der MS-DOS-Programmdateien verwirklicht. Weitere MS-DOS-Funktionen lassen sich über verschiedene Unterprogrammaufrufe ausführen. Folgende MS-DOS-Programmdateien müssen im gleichen Unterverzeichnis abgelegt sein, in dem sich auch Easydos befindet:

```
command.com
diskcopy.com
format.com
chkdsk.com
```

Bei entsprechender Korrektur der Unit »extern.pas« (Listing 4) können Sie natürlich die Exe-Versionen dieser Programmdateien ebenfalls verwenden. Beachten Sie dabei die Hinweise im Listing. Der Menüpunkt »Extern-Pfad setzen« gewährleistet auch dann eine korrekte Arbeitsweise, wenn Sie diese Programmdateien in anderen Verzeichnissen ablegen.

Easydos ist in der vorliegenden Version für Computersysteme mit nur einem Diskettenlaufwerk ausgelegt. Durch Änderung der Anweisungen für die bedingte Compilierung in den Units »general.pas« und »disk.pas« (siehe Hinweise in den Listings 2 und 3) läßt sich das Programm aber auch für Systeme mit zwei Diskettenlaufwerken anpassen.

(Karlheinz Büker/zi)

Name	global.pas, general.pas, disk.pas, extern.pas, easydos.pas
Sprache	Turbo Pascal 4.0
Grafikkarte	jede
Besonderheiten	keine

Aktivierung mit	Art der Funktion	Erläuterung
[F1]	System-Informationen	Ausgabe einiger wichtiger Informationen über die Systemkonfiguration.
[F2]	MS-DOS-Version ausgeben	Entspricht dem MS-DOS-Befehl »ver«.
[F3]	Verify-Status setzen	Entspricht dem MS-DOS-Kommando »verify«.
[F4]	Diskette formatieren	Menügesteuerter Aufruf von »format.com« (beziehungsweise »format.exe«).
[F5]	Diskette kopieren	Menügesteuerter Aufruf von »diskcopy.com« (beziehungsweise »diskcopy.exe«).
[F6]	Diskette prüfen	Menügesteuerter Aufruf von »chkdsk.com« (beziehungsweise »chkdsk.exe«).
[F7]	Programm starten	Startet jedes beliebige Programm, gegebenenfalls mit Parameterübergabe. Die Funktion prüft selbständig, ob das aufgerufene Programm lauffähig ist.

Aktivierung mit	Art der Funktion	Erläuterung
[F8]	Systemzeit setzen	Entspricht dem MS-DOS-Befehl »time« (menügesteuert).
[F9]	Systemdatum setzen	Entspricht dem MS-DOS-Befehl »date« (menügesteuert).
[F10]	Druckereinstellung	Stellt den Drucker (standardmäßig »lpt1«) ein. Alle wichtigen Einstellungen sind implementiert (46 verschiedene Parameter). Die Steuersequenzen beziehen sich auf den Schneider-Drucker DMP 3160. Eventuell sind hier entsprechende Änderungen vorzunehmen.
[Ctrl-F1]	Verzeichnis anzeigen	Entspricht dem MS-DOS-Kommando »dir«, ist jedoch wesentlich komfortabler. Die Ausgabe erfolgt mit Name, Zeit, Datum, Größe und Attributen. Die Filter »sort« und »more« sind bereits integriert. Eine Sortierung nach Name, Erweiterung, Zeit, Größe und Attributen ist auf Tastendruck durchführbar.
[Ctrl-F2]	Verzeichnis durchsuchen	Mit dieser Funktion läßt sich der gesamte Verzeichnisbaum auf Tastendruck durchsuchen. Mit den Cursor-Tasten wird ein Verzeichnis ausgewählt und mit [Enter] geöffnet. Wie unter MS-DOS gelangt man über Anwahl des Verzeichnisses ».« ins Stammverzeichnis zurück beziehungsweise über »..« ins jeweils übergeordnete Verzeichnis. Eine besonders für Festplattenbesitzer hilfreiche Funktion.
[Ctrl-F3]	Verzeichnis wechseln	Entspricht dem MS-DOS-Befehl »cd« (»chdir«). Alle verfügbaren Verzeichnisse werden vorgegeben.
[Ctrl-F4]	Verzeichnis anlegen	Entspricht dem MS-DOS-Kommando »md« (»mkdir«).
[Ctrl-F5]	Verzeichnis löschen	Entspricht dem MS-DOS-Befehl »rd« (»rmdir«). Alle verfügbaren Verzeichnisse werden vorgegeben.
[Ctrl-F6]	Verstecktes Verzeichnis anlegen	Ähnlich wie [Ctrl-F4]. Die mit dieser Funktion angelegten Verzeichnisse werden im MS-DOS-Modus allerdings nicht mehr angezeigt. Es ist ratsam, sich Namen und Lage der versteckten Verzeichnisse gut zu merken. »Easydos« behandelt sie allerdings genauso wie ihre sichtbaren Äquivalente.
[Ctrl-F7]	Verzeichnisstruktur	Entspricht im Prinzip dem MS-DOS-Befehl »tree«. Es werden alle verfügbaren Verzeichnisse ausgegeben, auf Wunsch mit oder ohne Anzeige der enthaltenen Dateien.
[Ctrl-F8]	MS-DOS-Kommandozeile	Über diese Funktion übernimmt »Easydos« alle Kommandozeilen, die auch MS-DOS akzeptieren würde.
[Ctrl-F9]	Extern-Pfad setzen	Hier kann ein Pfad vorgegeben werden, wenn die benötigten MS-DOS-Programmdateien »command«, »diskcopy«, »format« und »chkdsk« nicht im selben Verzeichnis wie »Easydos« stehen. Es wird dann der hier abgelegte Suchpfad verwendet.
[Ctrl-F10]	Zum MS-DOS-Modus	Lädt den Kommandointerpreter »command.com«. Durch das MS-DOS-Kommando »exit« erfolgt der Rücksprung zu »Easydos«.
[Shift-F1]	Dateinamen ändern	Entspricht dem MS-DOS-Befehl »ren« (»rename«).
[Shift-F2]	Datei löschen	Entspricht dem MS-DOS-Kommando »del«.
[Shift-F3]	Textdatei anzeigen	Entspricht dem MS-DOS-Befehl »type«, jedoch mit integriertem More-Filter.
[Shift-F4]	Datei kopieren	Entspricht dem MS-DOS-Befehl »copy«.
[Shift-F5]	Dateiattribute setzen	Alle Dateiattribute lassen sich menügesteuert manipulieren.
[Shift-F6]	Datei verschieben	Quelle und Ziel **müssen** auf einem Laufwerk liegen. Diese Funktion manipuliert lediglich den Verzeichniseintrag der jeweiligen Datei. Die Datei selbst bleibt völlig unverändert. Es sind folgende Unterfunktionen realisiert worden: 1) Nur umbenennen (entspricht [Shift-F1]) – Eingabe identischer Pfade mit unterschiedlichen Dateinamen 2) Nur verschieben – Eingabe unterschiedlicher Pfade mit identischen Dateinamen. Die Datei befindet sich anschließend im neuen Verzeichnis. 3) Umbenennen und Verschieben – Eingabe unterschiedlicher Verzeichnisse und unterschiedlicher Dateinamen. (Im Gegensatz zum Copy-Befehl bleibt an der alten Stelle im Verzeichnis kein Dateieintrag zurück. Physikalisch liegt die Datei aber immer noch an derselben Stelle auf dem Datenträger.)
[Shift-F7]	Dateizeit ändern	Die Zeitangabe der letzten Änderung einer Datei läßt sich manipulieren. Die alten Angaben werden angezeigt.
[Shift-F8]	Dateidatum ändern	Manipulation des Dateidatums analog zu [Shift-F7].
[Shift-F9]	Datei drucken	Gibt eine Datei auf dem angeschlossenen Drucker (standardmäßig »lpt1«) aus. Die Ausgabe erfolgt wahlweise mit oder ohne vorangestellte Zeilennummern. Der Ausdruck mit Zeilennummern ist vorwiegend für den Ausdruck von Programmlistings gedacht. Die Einstellung des Druckers erfolgt mit Funktion F10.
[Shift-F10]	Easydos beenden	Beendet »Easydos« und kehrt zur MS-DOS-Kommandozeile zurück.

Tabelle. Übersicht der Funktionen, die Ihnen »Easydos« zur Verfügung stellt

```
[Z35]    1: { Programmname: GLOBAL.PAS   }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[TW2]    5: {  U N I T   G L O B A L
[X24]    6:    Autor: Karlheinz Büker
[ØØØ]    7:
[GL6]    8:    Enthält Prozeduren und Funktionen, die
[D86]    9:    in allen Programmteilen von EASYDOS
[WU4]   1Ø:    benötigt werden.                        }
[ØØØ]   11:
[K22]   12: Unit Global;
[ØØØ]   13:
[SK1]   14: INTERFACE
[ØØØ]   15:
[ASØ]   16: Uses
[8G1]   17:   Dos, Crt;
[ØØØ]   18:
[RNØ]   19: Var
[EG1]   2Ø:   a    : Char;
[MG1]   21:   f    : File;
[9N1]   22:   t    : Text;
[ØØØ]   23:
[VY5]   24: Procedure WriteXY (x, y : Byte; Txt:
[3C1]   25:                    String);
[8G5]   26: { Kombination aus GotoXY+Write }
[8P6]   27: Procedure TextColors (Fore, Back : Byte);
[NK4]   28: { Textfarben setzen             }
[V33]   29: Procedure AktPath;
[CZ4]   3Ø: { gesetzten Pfad ausgeben       }
[UR3]   31: Procedure FullWindow;
[WX4]   32: { Fenster = kompl. Monitor      }
[AK3]   33: Procedure CursorOff;
[FL4]   34: { Cursor ausschalten            }
[VF3]   35: Procedure CursorOn;
[6K4]   36: { Cursor einschalten            }
[YW2]   37: Procedure Start;
[ØD4]   38: { Einschaltmeldung              }
[AG6]   39: Procedure xWindow (Titel : String; x1, y1,
[762]   4Ø:                    x2, y2 : Byte);
[ØØØ]   41:
[2D5]   42: { Fenster mit Rahmen und Titel }
[ØN2]   43: Procedure Ende;
[UF5]   44: { Funktionsende vorbereiten     }
[ML6]   45: Function  FindFile (Path : String) : Byte;
[VS4]   46: { Datei finden und testen       }
[CØ6]   47: Procedure Fehlermeldung (Nr : Byte);
[P15]   48: { Fehlermeldungen ausgeben      }
[ØØØ]   49:
[ØØØ]   5Ø:
[4L2]   51: IMPLEMENTATION
[ØØØ]   52:
[ØØØ]   53:
[HC6]   54: {--- Prüft Datei auf Vorhandensein und
[3R5]   55:      stellt fest ob lauffähig -----------}
[ØØØ]   56:
[W43]   57: Function FindFile;
[ØUØ]   58: Label
[RE1]   59:   Ausgang;
[MUØ]   6Ø: Begin
[ØH2]   61:   Assign (f,Path);
[DXØ]   62:   {$I-}
[NF1]   63:     Reset (f);
[BXØ]   64:   {$I+}
[263]   65:   If IOResult = Ø Then
[MUØ]   66:   Begin
[CE1]   67:     Close (f);
[K62]   68:     FindFile := Ø;
[7H4]   69:    { Path OK, Datei gefunden }
[A82]   7Ø:     Goto Ausgang;
[ZNØ]   71:   End;
[RL3]   72:   Assign (f,Path+'.COM');
[DXØ]   73:   {$I-}
[NF1]   74:     Reset (f);
[BXØ]   75:   {$I+}
[263]   76:   If IOResult = Ø Then
[MUØ]   77:   Begin
[CE1]   78:     Close (f);
[M62]   79:     FindFile := 1;
[LC5]   8Ø:    { Path OK, Datei ist COM Format }
[A82]   81:     Goto Ausgang;
[ZNØ]   82:   End;
[TM3]   83:   Assign (f,Path+'.EXE');
[DXØ]   84:   {$I-}
[NF1]   85:     Reset (f);
[BXØ]   86:   {$I+}
[263]   87:   If IOResult = Ø Then
[MUØ]   88:   Begin
[CE1]   89:     Close (f);
[P62]   9Ø:     FindFile := 2;
[ND5]   91:    { Path OK, Datei ist EXE Format }
[A82]   92:     Goto Ausgang;
[ZNØ]   93:   End;
[3K3]   94:   Assign (f,Path+'.BAT');
[DXØ]   95:   {$I-}
[NF1]   96:     Reset (f);
[BXØ]   97:   {$I+}
[263]   98:   If IOResult = Ø Then
[MUØ]   99:   Begin
[CE1]  1ØØ:     Close (f);
[S62]  1Ø1:     FindFile := 3;
[ZA5]  1Ø2:    { Path OK, Datei ist BAT Format }
[A82]  1Ø3:     Goto Ausgang;
[ZNØ]  1Ø4:   End;
[472]  1Ø5:   FindFile := 9;
[VV4]  1Ø6:  { Datei existiert nicht oder
[574]  1Ø7:    ist nicht lauffähig        }
[PE1]  1Ø8: Ausgang:
[ZNØ]  1Ø9: End;
[ØØØ]  11Ø:
[SM7]  111: {-- Bereitet Monitor für Hauptmenü vor --}
[ØØØ]  112:
[ØN2]  113: Procedure Ende;
[MUØ]  114: Begin
[V42]  115:   FullWindow;
[S64]  116:   TextColors (White,Black);
[DB1]  117:   ClrScr;
[ZNØ]  118: End;
[ØØØ]  119:
[7G7]  12Ø: {-- Schreibt Text an Bildschirmposition --}
[ØØØ]  121:
[683]  122: Procedure WriteXY;
[MUØ]  123: Begin
[M52]  124:   GotoXY (x,y);
[5W1]  125:   Write (Txt);
[ZNØ]  126: End;
[ØØØ]  127:
[9E6]  128: {-- Setzt Vorder- und Hintergrundfarbe
[G65]  129:     im Textmodus ------------------------}
[ØØØ]  13Ø:
[DT3]  131: Procedure TextColors;
[MUØ]  132: Begin
[EU2]  133:   TextColor (Fore);
[FM3]  134:   TextBackGround (Back);
[ZNØ]  135: End;
[ØØØ]  136:
[AE5]  137: {--- Gibt aktuelles Laufwerk und
[A56]  138:     gesetzten Pfad in Statuszeile aus --}
[ØØØ]  139:
[V33]  14Ø: Procedure AktPath;
[RNØ]  141: Var
[984]  142:   DefaultPath : String[64];
[MUØ]  143: Begin
[GT3]  144:   GetDir (Ø,DefaultPath);
[8K5]  145:   WriteXY ( 1,25,' Gesetzter Pfad : ');
[G84]  146:   TextColors (Black,White);
[GU3]  147:   Write ('  ',DefaultPath);
[S64]  148:   TextColors (White,Black);
[ZNØ]  149: End;
[ØØØ]  15Ø:
[845]  151: {--- Setzt Fenster auf maximale
[225]  152:     Koordinaten ------------------------}
[ØØØ]  153:
[UR3]  154: Procedure FullWindow;
[MUØ]  155: Begin
[TP2]  156:   Window (1,1,8Ø,25);
[ZNØ]  157: End;
[ØØØ]  158:
[376]  159: {--- Schaltet Cursor aus ----------------}
[ØØØ]  16Ø:
[AK3]  161: Procedure CursorOff;
[RNØ]  162: Var
[ØT2]  163:   Reg : Registers;
[MUØ]  164: Begin
[AN1]  165:   Reg.AH := 1;
[3T1]  166:   Reg.CH := 32;
[FT1]  167:   Reg.CL := 32;
[W32]  168:   Intr (16,Reg);
[ZNØ]  169: End;
[ØØØ]  17Ø:
[U56]  171: {--- Schaltet Cursor ein ----------------}
[ØØØ]  172:
[VF3]  173: Procedure CursorOn;
[RNØ]  174: Var
[ØT2]  175:   Reg : Registers;
[MUØ]  176: Begin
[AN1]  177:   Reg.AH := 1;
[2P1]  178:   Reg.CH := 6;
[KP1]  179:   Reg.CL := 7;
[W32]  18Ø:   Intr (16,Reg);
[ZNØ]  181: End;
[ØØØ]  182:
[286]  183: {--- Gibt Startmeldung aus --------------}
[ØØØ]  184:
[YW2]  185: Procedure Start;
[MUØ]  186: Begin
[LV1]  187:   CursorOff;
[VR2]  188:   Window (2Ø,8,6Ø,18);
[6D4]  189:   TextColors (Blue,Yellow);
[DB1]  19Ø:   ClrScr;
[472]  191:   WriteXY (8, 3,
[3L4]  192:            '(C) 1989  DOS International');
[872]  193:   WriteXY (8, 5,
[TD3]  194:           '   E A S Y D O S  1.1Ø    ');
[C72]  195:   WriteXY (8, 7,
[S85]  196:           ' MS-DOS-Benutzeroberfläche ');
[G72]  197:   WriteXY (8, 9,
[HZ4]  198:           '  Autor: Karlheinz Büker   ');
[YR1]  199:   Delay (5ØØØ);
[S64]  2ØØ:   TextColors (White,Black);
```

```
[ZNØ]  2Ø1: End;
[ØØØ]  2Ø2:
[3B6]  2Ø3: {--- Öffnet Fenster, zeichnet Rahmen
[3E6]  2Ø4:      und gibt Fensterüberschrift aus ----}
[ØØØ]  2Ø5:
[C73]  2Ø6: Procedure xWindow;
[RNØ]  2Ø7: Var
[4C1]  2Ø8:   a : Byte;
[MUØ]  2Ø9: Begin
[Y63]  21Ø:   Window (x1,y1,x2,y2);
[233]  211:   WriteXY ( 1, 1,'╒');
[X63]  212:   For a := 2 To x2-x1 Do
[TØ3]  213:     WriteXY (a, 1,'═');
[HC4]  214:   WriteXY (x2 - x1 + 1, 1,'╕');
[XL3]  215:   For a := 2 To y2 - y1 - 1 Do
[MUØ]  216:   Begin
[3Z2]  217:     WriteXY ( 1,a,'│');
[974]  218:     WriteXY (x2 - x1 + 1,a,'│');
[ZNØ]  219:   End;
[3Z3]  22Ø:   WriteXY ( 1,y2 - y1,'╘');
[273]  221:   For a := 2 To x2 - x1 Do
[GR3]  222:     WriteXY (a,y2 - y1,'═');
[GD5]  223:   WriteXY (x2 - x1 + 1,y2 - y1,'╛');
[ØA3]  224:   a := ((x2 - x1 - 2) Div 2) -
[Ø44]  225:        (Length (Titel) Div 2) + 3;
[G84]  226:   TextColors (Black,White);
[UE3]  227:   WriteXY (a, 1,Titel);
[S64]  228:   TextColors (White,Black);
[ZNØ]  229: End;
[ØØØ]  23Ø:
[VB6]  231: {--- Gibt Fehlermeldungen aus ----------}
[ØØØ]  232:
[U74]  233: Procedure Fehlermeldung;
[MUØ]  234: Begin
[2T2]  235:   Window (1,25,79,25);
[DB1]  236:   ClrScr;
[F72]  237:   GotoXY ( 3, 1);
[LB4]  238:   Write ('F E H L E R  : '#7#7#7#7);
[WP1]  239:   Case Nr Of
[XN5]  24Ø:     1 : Write ('DATEI ODER VERZEICHNIS '+
[N13]  241:                'NICHT GEFUNDEN !');
[LS5]  242:     2 : Write ('VERZEICHNIS EXISTIERT '+
[TL1]  243:                'SCHON !');
[4F5]  244:     3 : Write ('UNGÜLTIGER PFAD ODER '+
[ZV3]  245:                'DIRECTORY EXISTIERT '+
[TL1]  246:                'SCHON !');
[GZ4]  247:     4 : Write ('VERZEICHNIS NICHT '+
[GØ2]  248:                'GEFUNDEN !');
[KH5]  249:     5 : Write ('DATEI NICHT GEFUNDEN !');
[KP5]  25Ø:     6 : Write ('ATTRIBUT KONNTE NICHT '+
[W53]  251:                'GESETZT WERDEN !');
[746]  252:     7 : Write ('SYSTEMZEIT KONNTE NICHT '+
[W53]  253:                'GESETZT WERDEN !');
[H96]  254:     8 : Write ('SYSTEMDATUM KONNTE NICHT '+
[W53]  255:                'GESETZT WERDEN !');
[WM5]  256:     9 : Write ('COMMAND.COM - PROGRAMM'+
[PV3]  257:                'DATEI NICHT GEFUNDEN !');
[3Y5]  258:    1Ø : Write ('PARAMETER INKOMPATIBEL '+
[MP2]  259:                'ODER PROGRAMM-'+
[DP2]  26Ø:                'ABLAUFFEHLER !');
[776]  261:    11 : Write ('QUELL- UND ZIELLAUFWERK '+
[RT3]  262:                'IDENTISCH, FUNKTION '+
[5F2]  263:                'ABGEBROCHEN !');
[ED5]  264:    12 : Write ('DATEINAME EXISTIERT '+
[ØH3]  265:                'BEREITS IM ZIELVER'+
[C22]  266:                'ZEICHNIS !');
[8V5]  267:    13 : Write ('BEIDE ANGABEN IDENTISCH, '+
[NP2]  268:                'FUNKTION WIRD '+
[5F2]  269:                'ABGEBROCHEN !');
[RN5]  27Ø:    14 : Write ('PFADANGABE FEHLERHAFT '+
[ØF4]  271:                'ODER ENTHÄLT KEINE DOS-'+
[9B2]  272:                'PROGRAMME !');
[C85]  273:    15 : Write ('DOS - PROGRAMMDATEI '+
[N13]  274:                'NICHT GEFUNDEN !');
[2A5]  275:    16 : Write ('DATEI NICHT GEFUNDEN '+
[L84]  276:                'ODER NICHT LAUFFÄHIG !');
[MZ5]  277:    17 : Write ('PROGRAMMNAME DARF KEINE '+
[MZ3]  278:                'EXTENSION ENTHALTEN !');
[KØ5]  279:    18 : Write ('DRUCKER IST NICHT '+
[E43]  28Ø:                'BETRIEBSBEREIT !');
[Ø16]  281:    19 : Write ('DIESE PROGRAMMFUNKTION '+
[3V3]  282:                'UNTERSTÜTZT KEIN B-'+
[612]  283:                'LAUFWERK !');
[3C5]  284:    2Ø : Write ('FÜR COMPUTER MIT 2 '+
[WD4]  285:                'LAUFWERKEN GGF. EASYDOS '+
[773]  286:                'NEU COMPILIEREN !');
[ZNØ]  287:   End;
[WR1]  288:   Delay (3ØØØ);
[DB1]  289:   ClrScr;
[ZNØ]  29Ø: End;
[ØØØ]  291:
[KNØ]  292: End.
```

Listing 1. Die Unit »global.pas« definiert allgemeine Prozeduren und Funktionen

```
[MB5]    1: { Programmname: GENERAL.PAS  }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[6A2]    5: {$DEFINE LW1}
[WK5]    6: { Zeile weglassen, wenn mit zwei
[9R4]    7:   Floppys gearbeitet wird !!! }
[ØØØ]    8:
[Y63]    9: {  U N I T   G E N E R A L
[X24]   1Ø:    Autor: Karlheinz Büker
[ØØØ]   11:
[7Y5]   12:    Enthält Prozeduren und Funktionen
[GG6]   13:    zu allgemeienen Programmfunktionen  }
[ØØØ]   14:
[962]   15: Unit General;
[ØØØ]   16:
[SK1]   17: INTERFACE
[ØØØ]   18:
[ASØ]   19: Uses
[L74]   2Ø:   Dos, Crt, Printer, Global;
[ØØØ]   21:
[BD4]   22: Procedure SetSystemTime;
[FT4]   23: { Systemzeit setzen              }
[ØD4]   24: Procedure SetSystemDate;
[BU4]   25: { Systemdatum setzen             }
[E73]   26: Procedure SysInfo;
[5X5]   27: { System-Informationen ausgeben }
[7B4]   28: Procedure GetDosVersion;
[2M5]   29: { Dos-Versionsnummer ausgeben    }
[YS4]   3Ø: Procedure SetVerifyStatus;
[745]   31: { Verify-Status setzen           }
[CF4]   32: Procedure InstallPrinter;
[MB5]   33: { Drucker-Parameter setzen       }
[ØØØ]   34:
[ØØØ]   35:
[4L2]   36: IMPLEMENTATION
[ØØØ]   37:
[ZD6]   38: {--- Drucker-Einstellungen --------------}
[ØØØ]   39:
[CF4]   4Ø: Procedure InstallPrinter;
[RNØ]   41: Var
[323]   42:   Reg    : Registers;
[NTØ]   43:   Wahl,
[T42]   44:   Param  : Byte;
[MUØ]   45: Begin
[DB1]   46:   ClrScr;
[S45]   47:   xWindow (' D R U C K E R E I N S T E '+
[GA3]   48:            'L L U N G E N ',1,1,8Ø,25);
[A42]   49:   Reg.AX := $Ø2ØØ;
[WP1]   5Ø:   Reg.DX := Ø;
[X32]   51:   Intr (23,Reg);
[174]   52:   If (Reg.AX And $8ØØ) <> Ø Then
[MUØ]   53:   Begin
[W23]   54:     Fehlermeldung (18);
[2TØ]   55:     Ende;
[ØWØ]   56:     Exit;
[ZNØ]   57:   End;
[MN1]   58:   CursorOn;
[MR2]   59:   Window (2,2,79,23);
[G84]   6Ø:   TextColors (Black,White);
[ML5]   61:   WriteXY ( 2, 1,' Drucker-Reset          '+
[UT1]   62:                 '  <1>                 ');
[Z25]   63:   WriteXY ( 2, 3,' Elite                  EIN'+
[5B2]   64:                 '  <2>   AUS  <3>');
[3Y5]   65:   WriteXY ( 2, 4,' Proportional           EIN'+
[BB2]   66:                 '  <4>   AUS  <5>');
[Ø85]   67:   WriteXY ( 2, 5,' Schmal                 EIN'+
[HB2]   68:                 '  <6>   AUS  <7>');
[4S5]   69:   WriteXY ( 2, 6,' NLQ Standard           EIN'+
[RB2]   7Ø:                 '  <8>   AUS  <9>');
[R96]   71:   WriteXY ( 2, 7,' NLQ Proportional EIN'+
[CF2]   72:                 '  <1Ø>   AUS <11>');
[XX5]   73:   WriteXY ( 2, 9,' Tiefstellung           EIN'+
[KF2]   74:                 '  <12>   AUS <13>');
[UX5]   75:   WriteXY ( 2,1Ø,' Hochstellung           EIN'+
[SF2]   76:                 '  <14>   AUS <15>');
[HS5]   77:   WriteXY ( 2,11,' Doppeldruck            EIN'+
[YF2]   78:                 '  <16>   AUS <17>');
[936]   79:   WriteXY ( 2,12,' Kursivschrift          EIN'+
[4G2]   8Ø:                 '  <18>   AUS <19>');
[MH5]   81:   WriteXY ( 2,13,' Fettdruck              EIN'+
[HF2]   82:                 '  <2Ø>   AUS <21>');
[176]   83:   WriteXY ( 2,14,' Unterstreichen         EIN'+
[RF2]   84:                 '  <22>   AUS <23>');
[H16]   85:   WriteXY ( 2,15,' Doppelte Breite        EIN'+
[XF2]   86:                 '  <24>   AUS <25>');
[FT5]   87:   WriteXY ( 2,17,' Wagenrücklauf          '+
[AX1]   88:                 '  <26>                ');
[KY5]   89:   WriteXY ( 2,18,' Zeilenvorschub         '+
[CX1]   9Ø:                 '  <27>                ');
[D75]   91:   WriteXY ( 2,19,' Backspace              '+
[EX1]   92:                 '  <28>                ');
[RX5]   93:   WriteXY ( 2,2Ø,' Seitenvorschub         '+
[GX1]   94:                 '  <29>                ');
[ZE5]   95:   WriteXY (41, 1,' Linker Rand            '+
[ZW1]   96:                 '  <3Ø>                ');
[BF5]   97:   WriteXY (41, 2,' Rechter Rand           '+
[1X1]   98:                 '  <31>                ');
```

Listing 2. In der Unit »general.pas« sind allgemeine Programmroutinen enthalten

DOS-EXTRA
DATABOX

– Die Software zur DOS-EXTRA Nr. 7 –

auf 2 Disketten im Format 5 1/4“ oder
1 Diskette im Format 3 1/2“

sagenhaft günstig, extra viel gute Software fürs Geld!

DATABOX:

- mehr als der übliche Softwareservice
- bringt alle Programme der DOS EXTRA auf Datenträger

Für alle PCs unter MS-DOS als 5 1/4“- oder 3 1/2“- Diskette

Den Aufbau der Databox-Diskette finden Sie in der Datei »read.me« erklärt. Geben Sie deshalb vor dem Start »type read.me« ein.
Sämtliche Pascal- und C-Programme liegen auch in einer compilierten Form für Hercules- und Farbgrafikkarten vor. Es ist allerdings nicht gewährleistet, daß diese Versionen auf Hardware, die nicht vollständig kompatibel zum IBM PC ist, problemlos laufen. In diesem Fall müssen Sie den Quellcode nach der Anpassung mit dem entsprechenden Compiler neu übersetzen. Die Basic-Programme arbeiten mit dem GW-Basic-Interpreter von Microsoft einwandfrei. Bei Interpretern und Compilern anderer Hersteller kann es zu Problemen bei der Programmausführung kommen, sofern diese nicht voll kompatibel sind.

Databoxinhalte der DOS EXTRA Nr. 7

- MS-DOS-Benutzeroberfläche Easydos
- Benutzeroberfläche und Präprozessor für SBasic
- Datengenerator für Basic, C und Pascal
- Sprite-Editor für Farbgrafikkarten
- DOS-Packer
- Residenten Programmen auf der Spur
- Virenschutzprogramm
- Monitorprogramm zur Tastaturauswertung
- Verwaltung von Programm-Modulen in C
- Weltkarte im Computer
- Schneller Zugriff auf der CGA-Karte
- Kurven einfacher plotten
- Einfacher Prozessor-Emulator
- SCheck 2.01: Checksummer für geprüfte Eingaben
- sowie alle begleitenden Listings zu den Grundlagenbeiträgen und der Tips-&-Tricks-Rubrik

2 Disketten 5 1/4“ (Bestell-Nr. 3817)
1 Diskette 3 1/2“ (Bestell-Nr. 3818)

DM 35,– *

Noch erhältlich:

Databox DOS EXTRA Nr. 1

- SAM, der DOS-Assembler
- Clock, Uhr-Utilities
- Pascal – Include-Datei zum Aufruf von BIOS-Funktionen unter Turbo Pascal
- Tips und Tricks für jeden Zweck

Diskette 5 1/4“ (Bestell-Nr. 350)
Diskette 3 1/2“ (Bestell-Nr. 351)

DM 24,– *

Databox DOS EXTRA Nr. 2

- DOS-Commander, eine Benutzeroberfläche für MS-DOS
- SCopy, ein Kopierprogramm
- SForth 3.1, ein Compiler im Selbstbau
- SCAD, (3D-Grafik)
- Hurrican, ein Actionspiel
- Supermind, ein Strategiespiel
- Utilities, Routinen für die tägliche Arbeit
- SCheck – Checksummer für die Eingabe

4 Disketten 5 1/4“ (Bestell-Nr. 352)
2 Disketten 3 1/2“ (Bestell-Nr. 353)

DM 45,– *

Databox DOS EXTRA Nr. 3

- Datenfernübertragungsprogramm in 100% Assemblercode
- Hercules-Grafik-Toolbox in Turbo Pascal
- Grafikfunktionen für Turbo C
- Zeichensatzeditor für Epson-kompatible Drucker
- Treiber für AT-Tastatur
- Alle begleitenden Listings zum Kurs »PC-glasklar«
- Demo für Sprachdigitalisierung
- ...und alle weiteren Listings

2 Disketten 5 1/4“ (Bestell-Nr. 357)
1 Diskette 3 1/2“ (Bestell-Nr. 358)

DM 35,– *

Databox DOS EXTRA Nr. 4

- Desktop-Publishing-Programm »SDTP«
- Kommandoeditor für MS-DOS
- Editor für Disketten und Festplatten
- Speicherresidente Programme auflisten und entfernen
- Paßwort-Schutz für MS-DOS
- Tastatureingabe automatisiert
- Clock-Utility für alle Grafikkarten
- Cursor und Zeichensatz für EGA-Karte programmieren
- Fenstertechnik für Turbo C
- Generator für Hilfsbildschirme
- Geprüfte Eingabe mit »SCheck«

2 Disketten 5 1/4“ (Bestell-Nr. 3811)
1 Diskette 3 1/2“ (Bestell-Nr. 3812)

DM 35,– *

Databox DOS EXTRA Nr. 5

- 16 Farben für die CGA-Farbgrafikkarte
- Snapshot: Bildschirmfotos per Software
- Druckerzeichensatz selbstgemacht
- Mausgesteuertes Zeichenprogramm
- Editor in GW-Basic
- Setup-Programm für ATs
- Maschinensprache für GW-Basic
- Erklärtes Listing: das Basic-Spiel »Stardos«
- Neuer Checksummer für geprüfte Eingabe

2 Disketten 5 1/4“ (Bestell-Nr. 3813)
4 Disketten 3 1/2“ (Bestell-Nr. 3814)

DM 35,– *

Databox DOS EXTRA Nr. 6

- SBasic-Compiler im Quelltext
- Datenbankprogramm »Parat«
- Fensterorientierte Funktionen für GW-Basic und SBasic
- Funktionen zur Stringverwaltung für GW-Basic und SBasic
- Turbo-Basic-Bibliothek für 16 Farben auf der CGA-Karte
- Hilfsfunktion für Basic-Grafikbefehle
- Programm zur Analyse von Grafikkarten
- Preview-Funktion für Dateien
- Maskengenerator zum Erzeugen von Binärdateien
- dBase-Datenbanken in Basic weiterverarbeiten
- Beispiele für rekursive Programmierung
- Superhirn für Quick Basic 4.0
- SCheck 2.0: Checksummer für geprüfte Eingaben in Basic
- sowie alle begleitenden Listings zu den Rubriken Basic intern und Tips & Tricks

2 Disketten 5 1/4“ (Bestell-Nr. 3815)
1 Diskette 3 1/2“ (Bestell-Nr. 3816)

DM 35,– *

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 4,– DM bzw. für das Ausland 6,– DM Porto und Verpackung. – Unverbindliche Preisempfehlung – Bitte benutzen Sie die Bestellkarte.

DMV-Verlag · Postfach 250 · 3440 Eschwege

```
[1C6]   99:   WriteXY (41, 3,' Seitenlänge Zeilen  '+
[3X1]  100:                   ' <32>               ');
[846]  101:   WriteXY (41, 4,' Seitenlänge Zoll    '+
[5X1]  102:                   ' <33>               ');
[9D6]  103:   WriteXY (41, 5,' Perfor. überspr. EIN'+
[2G2]  104:                   ' <34>   AUS <35>');
[FØ6]  105:   WriteXY (41, 7,' Horiz. TAB setzen   '+
[BX1]  106:                   ' <36>               ');
[ES5]  107:   WriteXY (41, 8,' Zum horiz. TAB      '+
[DX1]  108:                   ' <37>               ');
[9P5]  109:   WriteXY (41, 9,' Vert. TAB setzen    '+
[FX1]  110:                   ' <38>               ');
[XF5]  111:   WriteXY (41,1Ø,' Zum vert. TAB       '+
[HX1]  112:                   ' <39>               ');
[UW5]  113:   WriteXY (41,12,' Zeilenabstand 1/8  "'+
[ØX1]  114:                   ' <4Ø>               ');
[8Z5]  115:   WriteXY (41,13,' Zeilenabstand 7/72 "'+
[2X1]  116:                   ' <41>               ');
[UW5]  117:   WriteXY (41,14,' Zeilenabstand 1/6  "'+
[4X1]  118:                   ' <42>               ');
[8Ø6]  119:   WriteXY (41,15,' Zeilenabstand n/216"'+
[6X1]  12Ø:                   ' <43>               ');
[8Ø6]  121:   WriteXY (41,16,' Zeilenabstand n/72 "'+
[8X1]  122:                   ' <44>               ');
[Z16]  123:   WriteXY (41,18,' Zeilenvorschub   n/'+
[S32]  124:   '216"  <45>          ');
[M66]  125:   WriteXY (41,19,' Zeilenrückschub  n/'+
[V32]  126:   '216"  <46>          ');
[S64]  127:   TextColors (White,Black);
[HH4]  128:   WriteXY (41,2Ø,'          < 99 >  =   '+
[R12]  129:   'ENDE            ');
[V36]  13Ø:   WriteXY (15,22,' INKOMPATIBLE PARAME'+
[Y34]  131:                   'TER ÜBERSTEUERN ALTE'+
[3Z2]  132:                   ' EINTELLUNGEN !');
[1S2]  133:   Window (1,25,8Ø,25);
[P31]  134:   Repeat
[DB1]  135:     ClrScr;
[H35]  136:     WriteXY ( 1, 1,'Bitte Funktion '+
[ZM2]  137:                    'auswählen  : ');
[P31]  138:     Repeat
[C82]  139:     GotoXY (3Ø, 1);
[DXØ]  14Ø:       {$I-}
[562]  141:         ReadLn (Wahl);
[BXØ]  142:       {$I+}
[9U3]  143:     Until (IOResult = Ø) And
[5V3]  144:           (((Wahl > Ø) And (Wahl < 47))
[N42]  145:           Or (Wahl = 99));
[5Z1]  146:     Case Wahl Of
[B63]  147:       1 : Write (Lst,#27'@');
[U63]  148:       2 : Write (Lst,#27'M');
[Z63]  149:       3 : Write (Lst,#27'P');
[6A3]  15Ø:       4 : Write (Lst,#27'p'#1);
[4A3]  151:       5 : Write (Lst,#27'p'#Ø);
[VS2]  152:       6 : Write (Lst,#15);
[6T2]  153:       7 : Write (Lst,#18);
[PA3]  154:       8 : Write (Lst,#27'x'#1);
[MA3]  155:       9 : Write (Lst,#27'x'#Ø);
[3H4]  156:      1Ø : Write (Lst,#27'x'#1#27'p'#1);
[4H4]  157:      11 : Write (Lst,#27'x'#Ø#27'p'#Ø);
[EL3]  158:      12 : Write (Lst,#27'S'#1);
[KC3]  159:      13 : Write (Lst,#27'T');
[LL3]  16Ø:      14 : Write (Lst,#27'S'#Ø);
[SC3]  161:      15 : Write (Lst,#27'T');
[1C3]  162:      16 : Write (Lst,#27'G');
[6C3]  163:      17 : Write (Lst,#27'H');
[1B3]  164:      18 : Write (Lst,#27'4');
[6B3]  165:      19 : Write (Lst,#27'5');
[DB3]  166:      2Ø : Write (Lst,#27'E');
[KB3]  167:      21 : Write (Lst,#27'F');
[4H3]  168:      22 : Write (Lst,#27'-'#1);
[6H3]  169:      23 : Write (Lst,#27'-'#Ø);
[4Ø3]  17Ø:      24 : Write (Lst,#14);
[UZ2]  171:      25 : Write (Lst,#2Ø);
[6Ø3]  172:      26 : Write (Lst,#13);
[XZ2]  173:      27 : Write (Lst,#1Ø);
[NW2]  174:      28 : Write (Lst,#8);
[BØ3]  175:      29 : Write (Lst,#12);
[DC1]  176:      3Ø : Begin
[DB1]  177:             ClrScr;
[DH4]  178:             WriteXY ( 1, 1,' Ab welcher '+
[ME4]  179:                'Druckposition drucken ? ');
[P31]  18Ø:             Repeat
[DXØ]  181:               {$I-}
[E12]  182:                 Read (Param);
[BXØ]  183:               {$I+}
[9U3]  184:             Until (IOResult = Ø) And
[Z72]  185:                   (Param >= Ø) And
[332]  186:                   (Param <= 255);
[2F4]  187:             Write (Lst,#27'l',Chr(param));
[ZNØ]  188:           End;
[GC1]  189:      31 : Begin
[DB1]  19Ø:             ClrScr;
[3P4]  191:             WriteXY ( 1, 1,' Bis welcher '+
[ME4]  192:                'Druckposition drucken ? ');
[P31]  193:             Repeat
[DXØ]  194:               {$I-}
[E12]  195:                 Read (Param);
[BXØ]  196:               {$I+}
[9U3]  197:             Until (IOResult = Ø) And
[D62]  198:                   (Param > Ø) And
[332]  199:                   (Param <= 255);
[7F4]  2ØØ:             Write (Lst,#27'Q',Chr(param));
[ZNØ]  2Ø1:           End;
[LC1]  2Ø2:      32 : Begin
[DB1]  2Ø3:             ClrScr;
[264]  2Ø4:             WriteXY ( 1, 1,' Wieviele '+
[KS3]  2Ø5:                'Zeilen pro Seite ? ');
[P31]  2Ø6:             Repeat
[DXØ]  2Ø7:               {$I-}
[E12]  2Ø8:                 Read (Param);
[BXØ]  2Ø9:               {$I+}
[9U3]  21Ø:             Until (IOResult = Ø) And
[D62]  211:                   (Param > Ø) And
[HW1]  212:                   (Param < 128);
[TE4]  213:             Write (Lst,#27'C',Chr(param));
[ZNØ]  214:           End;
[PC1]  215:      33 : Begin
[DB1]  216:             ClrScr;
[K34]  217:             WriteXY ( 1, 1,' Wieviel '+
[TY3]  218:                'Zoll Seitenlänge ? ');
[P31]  219:             Repeat
[DXØ]  22Ø:               {$I-}
[E12]  221:                 Read (Param);
[BXØ]  222:               {$I+}
[9U3]  223:             Until (IOResult = Ø) And
[D62]  224:                   (Param > Ø) And
[WR1]  225:                   (Param < 23);
[ØN4]  226:             Write (Lst,#27'C'#Ø,Chr(param))
       ;
[ZNØ]  227:           End;
[KL3]  228:      34 : Write (Lst,#27'N'#5);
[MA3]  229:      35 : Write (Lst,#27'Ø');
[2C1]  23Ø:      36 : Begin
[DB1]  231:             ClrScr;
[SB4]  232:             WriteXY ( 1, 1,' vertikale '+
[SH3]  233:                    'Tabulatorposition '+
[K72]  234:                    'eingeben : ');
[P31]  235:             Repeat
[DXØ]  236:               {$I-}
[E12]  237:                 Read (Param);
[BXØ]  238:               {$I+}
[9U3]  239:             Until (IOResult = Ø) And
[D62]  24Ø:                   (Param > Ø) And
[NW1]  241:                   (Param < 138);
[RR4]  242:             Write (Lst,#27'D',Chr(param),#Ø
       );
[ZNØ]  243:           End;
[RW2]  244:      37 : Write (Lst,#9);
[5D1]  245:      38 : Begin
[DB1]  246:             ClrScr;
[UG4]  247:             WriteXY (1,1,' Horizontale '+
[SH3]  248:                    'Tabulatorposition '+
[K72]  249:                    'eingeben : ');
[P31]  25Ø:             Repeat
[DXØ]  251:               {$I-}
[E12]  252:                 Read (Param);
[BXØ]  253:               {$I+}
[9U3]  254:             Until (IOResult = Ø) And
[D62]  255:                   (Param > Ø) And
[XW1]  256:                   (Param < 255);
[NR4]  257:             Write (Lst,#27'B',Chr(param),#Ø
       );
[ZNØ]  258:           End;
[9Ø3]  259:      39 : Write (Lst,#11);
[DY2]  26Ø:      4Ø : Write (Lst,'Ø');
[LY2]  261:      41 : Write (Lst,'1');
[TY2]  262:      42 : Write (Lst,'2');
[SC1]  263:      43 : Begin
[DB1]  264:             ClrScr;
[K34]  265:             WriteXY ( 1, 1,' Wieviel '+
[392]  266:                    '216tel Zoll '+
[233]  267:                    'Zeilenabstand ? ');
[P31]  268:             Repeat
[DXØ]  269:               {$I-}
[E12]  27Ø:                 Read (Param);
[BXØ]  271:               {$I+}
[9U3]  272:             Until (IOResult = Ø) And
[Z72]  273:                   (Param >= Ø) And
[YW1]  274:                   (Param < 256);
[9E4]  275:             Write (Lst,#27'3',Chr(param));
[ZNØ]  276:           End;
[VC1]  277:      44 : Begin
[DB1]  278:             ClrScr;
[K34]  279:             WriteXY ( 1, 1,' Wieviel '+
[SZ1]  28Ø:                    '72tel Zoll '+
[233]  281:                    'Zeilenabstand ? ');
[P31]  282:             Repeat
[DXØ]  283:               {$I-}
[E12]  284:                 Read (Param);
[BXØ]  285:               {$I+}
[9U3]  286:             Until (IOResult = Ø) And
[Z72]  287:                   (Param >= Ø) And
[US1]  288:                   (Param < 86);
[RE4]  289:             Write (Lst,#27'A',Chr(param));
[ZNØ]  29Ø:           End;
[YC1]  291:      45 : Begin
[DB1]  292:             ClrScr;
[K34]  293:             WriteXY ( 1, 1,' Wieviel '+
```

Listing 2. In der Unit »general.pas« sind allgemeine Programmroutinen enthalten

```
[392]  294:                      '216tel Zoll '+
[1D3]  295:                      'Zeilenvorschub ? ');
[P31]  296:              Repeat
[DXØ]  297:                {$I-}
[E12]  298:                  Read (Param);
[BXØ]  299:                {$I+}
[9U3]  3ØØ:              Until (IOResult = Ø) And
[Z72]  3Ø1:                    (Param >= Ø) And
[YW1]  3Ø2:                    (Param < 256);
[ØF4]  3Ø3:              Write (Lst,#27'J',Chr(param));
[ZNØ]  3Ø4:            End;
[1D1]  3Ø5:       46 : Begin
[DB1]  3Ø6:              ClrScr;
[K34]  3Ø7:              WriteXY ( 1, 1,' Wieviel '+
[392]  3Ø8:                      '216tel Zoll '+
[1P3]  3Ø9:                      'Zeilenrückschub ? ');
[P31]  31Ø:              Repeat
[DXØ]  311:                {$I-}
[E12]  312:                  Read (Param);
[BXØ]  313:                {$I+}
[9U3]  314:              Until (IOResult = Ø) And
[Z72]  315:                    (Param >= Ø) And
[YW1]  316:                    (Param < 256);
[ØF4]  317:              Write (Lst,#27'j',Chr(param));
[ZNØ]  318:            End;
[ZNØ]  319:       End;
[XF2]  32Ø:     Until wahl = 99;
[LV1]  321:     CursorOff;
[ZTØ]  322:     Ende;
[ZNØ]  323: End;
[ØØØ]  324:
[1A6]  325: {--- Systemzeit setzen ------------------}
[ØØØ]  326:
[BD4]  327: Procedure SetSystemTime;
[6XØ]  328: Const
[AP2]  329:   Sek1ØØ  : Word = Ø;
[RNØ]  33Ø: Var
[BA1]  331:   Stunde,
[991]  332:   Minute,
[DL2]  333:   Sekunde : Word;
[MUØ]  334: Begin
[DB1]  335:   ClrScr;
[5R4]  336:   xWindow (' S Y S T E M Z E I T   '+
[C23]  337:            'S E T Z E N ',1,1,8Ø,25);
[MN1]  338:   CursorOn;
[P31]  339:   Repeat
[R93]  34Ø:     XWindow ('',2Ø,7,6Ø,1Ø);
[A56]  341:     WriteXY (3,2,'Bitte Stunde eingeben '+
[2N1]  342:                  '(Ø - 23)  :  ');
[DXØ]  343:     {$I-}
[Z72]  344:     Read (Stunde);
[BXØ]  345:     {$I+}
[9U3]  346:   Until (IOResult = Ø) And
[7P4]  347:         (Stunde >= Ø) And (Stunde <= 23);
[P31]  348:   Repeat
[8E3]  349:     XWindow ('',2Ø,12,6Ø,15);
[VD6]  35Ø:     WriteXY (3,2,'Bitte Minuten eingeben '+
[VN1]  351:                  '(Ø - 59) :  ');
[DXØ]  352:     {$I-}
[392]  353:     Read (Minute);
[BXØ]  354:     {$I+}
[9U3]  355:   Until (IOResult = Ø) And
[WP4]  356:         (Minute >= Ø) And (Minute <= 59);
[P31]  357:     Repeat
[1E3]  358:     XWindow ('',2Ø,17,6Ø,2Ø);
[VC6]  359:     WriteXY (3,2,'Bitte Sekunden eingeben'+
[3M1]  36Ø:                  ' (Ø - 59):  ');
[DXØ]  361:     {$I-}
[6G2]  362:     Read (Sekunde);
[BXØ]  363:     {$I+}
[9U3]  364:   Until (IOResult = Ø) And
[U45]  365:         (Sekunde >= Ø) And (Sekunde <= 59);
[DXØ]  366:   {$I-}
[5L6]  367:   SetTime (Stunde,Minute,Sekunde,Sek1ØØ);
[BXØ]  368:   {$I+}
[XA6]  369:   If IOResult <> Ø Then Fehlermeldung (7);
[LV1]  37Ø:   CursorOff;
[ZTØ]  371:   Ende;
[ZNØ]  372: End;
[ØØØ]  373:
[HD6]  374: {--- Systemdatum setzen ------------------}
[ØØØ]  375:
[ØD4]  376: Procedure SetSystemDate;
[RNØ]  377: Var
[LNØ]  378:   Tag,
[C11]  379:   Monat,
[F12]  38Ø:   Jahr   : Word;
[MUØ]  381: Begin
[DB1]  382:   ClrScr;
[UT4]  383:   xWindow (' S Y S T E M D A T U M   '+
[C23]  384:            'S E T Z E N ',1,1,8Ø,25);
[MN1]  385:   CursorOn;
[P31]  386:   Repeat
[PA3]  387:     XWindow ('',17,7,63,1Ø);
[HM5]  388:     WriteXY (3,2,'Bitte Tag eingeben '+
[H32]  389:                  '(1 - 31)       :  ');
[DXØ]  39Ø:     {$I-}
[TR1]  391:     Read (Tag);
[BXØ]  392:     {$I+}
[9U3]  393:   Until (IOResult = Ø) And
[5N3]  394:         (Tag >= 1) And (Tag <= 31);
[P31]  395:   Repeat
[YE3]  396:     XWindow ('',17,12,63,15);
[GW5]  397:     WriteXY (3,2,'Bitte Monat eingeben '+
[8Z1]  398:                  '(1 - 12)      :  ');
[DXØ]  399:     {$I-}
[422]  4ØØ:     Read (Monat);
[BXØ]  4Ø1:     {$I+}
[9U3]  4Ø2:   Until (IOResult = Ø) And
[SF4]  4Ø3:         (Monat >= 1) And (Monat <= 12);
[P31]  4Ø4:   Repeat
[RE3]  4Ø5:     XWindow ('',17,17,63,2Ø);
[AN5]  4Ø6:     WriteXY (3,2,'Bitte Jahr eingeben '+
[162]  4Ø7:                  '(198Ø - 2Ø99) :  ');
[DXØ]  4Ø8:     {$I-}
[1U1]  4Ø9:     Read (Jahr);
[BXØ]  41Ø:     {$I+}
[9U3]  411:   Until (IOResult = Ø) And
[FG4]  412:         (Jahr >= 1Ø8Ø) And (Jahr <= 2Ø99);
[DXØ]  413:   {$I-}
[174]  414:   SetDate (Jahr,Monat,Tag);
[BXØ]  415:   {$I+}
[ØB6]  416:   If IOResult <> Ø Then Fehlermeldung (8);
[LV1]  417:   CursorOff;
[ZTØ]  418:   Ende;
[ZNØ]  419: End;
[ØØØ]  42Ø:
[KC6]  421: {--- VERIFY-Status setzen ---------------}
[ØØØ]  422:
[YS4]  423: Procedure SetVerifyStatus;
[RNØ]  424: Var
[ZM1]  425:   c      : Char;
[XS2]  426:   Verify : Boolean;
[323]  427:   Reg    : Registers;
[MUØ]  428: Begin
[V42]  429:   FullWindow;
[DB1]  43Ø:   ClrScr;
[HY5]  431:   xWindow (' VERIFY-STATUS ',25,5,55,1Ø);
[666]  432:  { aktuellen Verify-Status ausgeben }
[NT1]  433:   Reg.AH := 84;
[8T1]  434:   MsDos (Reg);
[A23]  435:   Verify := Reg.AL = 1;
[C82]  436:   GotoXY (1Ø, 3);
[436]  437:   If Verify Then Write ('Verify ist EIN')
[445]  438:     Else Write ('Verify ist AUS');
[9R5]  439:   xWindow (' VERIFY-Status setzen ',
[CH1]  44Ø:             25,15,55,21);
[7M4]  441:  { Status ggf. verändern }
[VZ3]  442:   WriteXY ( 2, 3,' CR        =  '+
[FØ3]  443:                'Status ändern');
[7L4]  444:   WriteXY ( 2, 4,' sonstige  =  '+
[FR2]  445:                'nicht  ändern');
[2Z1]  446:   c := ReadKey;
[712]  447:   If c = #13 Then
[MUØ]  448:   Begin
[7T1]  449:     Reg.AH := 46;
[TU3]  45Ø:     If Verify Then Reg.AL := Ø
[HK2]  451:     Else Reg.AL := 1;
[8T1]  452:     MsDos (Reg);
[ZNØ]  453:   End;
[ZTØ]  454:   Ende;
[ZNØ]  455: End;
[ØØØ]  456:
[9Y6]  457: {--- Dos-Versionsnummer ausgeben --------}
[ØØØ]  458:
[7B4]  459: Procedure GetDosVersion;
[RNØ]  46Ø: Var
[ØT2]  461:   Reg : Registers;
[MUØ]  462: Begin
[V42]  463:   FullWindow;
[DB1]  464:   ClrScr;
[FE3]  465:   xWindow ('',12,11,69,14);
[WP5]  466:  { DOS-Versionsnummer ausgeben }
[E72]  467:   GotoXY ( 2, 2);
[9T1]  468:   Reg.AH := 48;
[8T1]  469:   MsDos (Reg);
[G84]  47Ø:   TextColors (Black,White);
[KØ5]  471:   Write (' Dieses System wurde '+
[4F5]  472:          'installiert unter Dos-Version ',
[A53]  473:          Reg.AL,'.',Reg.AH,' ');
[YR1]  474:   Delay (5ØØØ);
[ZTØ]  475:   Ende;
[ZNØ]  476: End;
[ØØØ]  477:
[2Y6]  478: {--- System-Informationen ausgeben ------}
[ØØØ]  479:
[E73]  48Ø: Procedure SysInfo;
[RNØ]  481: Var
[SG2]  482:   c                  : Char;
[EU3]  483:   Reg                : Registers;
[L13]  484:   SektorenProBlock,
[3G2]  485:   BlocksProDisk,
[PE4]  486:   BlockGroesse       : Integer;
[XR3]  487:   Bytes              : LongInt;
[9ZØ]  488:   Break,
[4T3]  489:   Verify             : Boolean;
```

```
[E8Ø]  49Ø:    e,
[FA1]  491:    Intern,
[ØWØ]  492:    RS232,
[WHØ]  493:    NP,
[PR2]  494:    DD                  : Byte;
[MUØ]  495: Begin
[DB1]  496:    ClrScr;
[FE5]  497:    xWindow (' S Y S T Ξ M I N F O R M A T '+
[KU2]  498:             'I O N E N ',1,1,8Ø,25);
[ZP4]  499:    xWindow (' Speicher ',4,3,17,8);
[Y32]  5ØØ:    Intr (18,Reg);
[P72]  5Ø1:    GotoXY ( 4, 3);
[KH3]  5Ø2:    Write (Reg.AX,' KByte');
[X32]  5Ø3:    Intr (17,Reg);
[AZ5]  5Ø4:    xWindow (' Konfiguration ',51,3,78,1Ø);
[4S5]  5Ø5:   { Konfigurationsbyte auswerten }
[M72]  5Ø6:    GotoXY ( 4, 2);
[Y84]  5Ø7:    Write  ('Drucker Ports    : ',
[F82]  5Ø8:             Reg.AH Shr 6:2);
[P72]  5Ø9:    GotoXY ( 4, 3);
[LØ4]  51Ø:    Write  ('Game-Adapter     : ',
[TR3]  511:             (Reg.AH Shr 3) Div 128:2);
[AC3]  512:    RS232 := (Reg.AH Shl 4);
[S72]  513:    GotoXY ( 4, 4);
[MY3]  514:    Write  ('RS232-Ports      : ',
[D62]  515:             RS232 Shr 5:2);
[U72]  516:    GotoXY ( 4, 5);
[LU2]  517:    DD := Reg.AL Shl 7;
[FF4]  518:    DD := (DD Shr 7) + (Reg.AL Shr 6);
[HB5]  519:    Write  ('Disk-Laufwerke   : ',DD:2);
[W72]  52Ø:    GotoXY ( 4, 6);
[EW2]  521:    NP := Reg.AL Shr 6;
[492]  522:    NP := NP Shr 7;
[MØ2]  523:    If NP = 1 Then
[AZ4]  524:      Write ('Num.-Coprozessor: Ja')
[W96]  525:    Else Write ('Num.-Coprozessor: Nein');
[285]  526:    xWindow (' Standard-Parameter ',
[ED1]  527:             2Ø,3,48,8);
[6L3]  528:   { Status anzeigen }
[NT1]  529:    Reg.AH := 84;
[8T1]  53Ø:    MsDos (Reg);
[A23]  531:    Verify := Reg.AL = 1;
[M72]  532:    GotoXY ( 4, 2);
[2A2]  533:    If Verify Then
[MZ3]  534:      Write ('Verify          : EIN')
[T15]  535:    Else Write ('Verify          : AUS');
[6T1]  536:    Reg.AH := 51;
[KN1]  537:    Reg.AL := Ø;
[8T1]  538:    MsDos (Reg);
[CY2]  539:    Break:= Reg.DL = 1;
[P72]  54Ø:    GotoXY ( 4, 3);
[512]  541:    If Break Then
[NB4]  542:      Write ('Break (CTRL-C) : EIN')
[4E5]  543:    Else Write ('Break (CTRL-C) : AUS');
[P55]  544:    xWindow (' Standard-Laufwerk ',
[LD1]  545:             3,9,48,2Ø);
[2R4]  546:   { Infos über Standard-LW }
[YS1]  547:    Reg.AH := 25;
[8T1]  548:    MsDos (Reg);
[YM2]  549:    Intern := Reg.AL;
[P72]  55Ø:    GotoXY ( 4, 3);
[ZD5]  551:    Write  ('Standard - Laufwerk      : ',
[L83]  552:            Chr ((Reg.AL) + 65),':');
[YØ3]  553:    Reg.AL := intern + 1;
[9T1]  554:    Reg.AH := 54;
[8T1]  555:    MsDos (Reg);
[S72]  556:    GotoXY ( 4, 4);
[9L3]  557:    If Not (Reg.AX = -1) Then
[MUØ]  558:    Begin
[YY4]  559:      Write ('Blockgröße              : ',
[YM2]  56Ø:             Reg.CX,' Bytes');
[P14]  561:      BlockGroesse := Reg.CX;
[4MØ]  562:    End
[ZRØ]  563:    Else
[MUØ]  564:    Begin
[WY4]  565:      Write ('Blockgröße              : '+
[3P3]  566:             'nicht feststellbar');
[ZNØ]  567:    End;
[413]  568:    Reg.DL := Intern + 1;
[9T1]  569:    Reg.AH := 54;
[8T1]  57Ø:    MsDos (Reg);
[U72]  571:    GotoXY ( 4, 5);
[9L3]  572:    If Not (Reg.AX = -1) Then
[MUØ]  573:    Begin
[S95]  574:      Write ('Sektoren pro Block      : ',
[691]  575:             Reg.AX);
[GS4]  576:      SektorenProBlock := Reg.AX;
[4MØ]  577:    End
[ZRØ]  578:    Else
[MUØ]  579:    Begin
[P95]  58Ø:      Write ('Sektoren Pro Block      : '+
[3P3]  581:             'nicht feststellbar');
[ZNØ]  582:    End;
[413]  583:    Reg.DL := Intern + 1;
[9T1]  584:    Reg.AH := 54;
[8T1]  585:    MsDos (Reg);
[W72]  586:    GotoXY ( 4, 6);
[9L3]  587:    If Not (Reg.AX = -1) Then
[MUØ]  588:    Begin
[3D5]  589:      Write ('Gesamt Blocks auf Disk   : ',
[C91]  59Ø:             Reg.DX);
[S24]  591:      BlocksProDisk := Reg.DX;
[4MØ]  592:    End
[ZRØ]  593:    Else
[MUØ]  594:    Begin
[1D5]  595:      Write ('Gesamt Blocks auf Disk   : '+
[3P3]  596:             'nicht feststellbar');
[ZNØ]  597:    End;
[Y72]  598:    GotoXY ( 4, 7);
[L46]  599:    Write  ('Gesamte Disk-Kapazität   : ',
[LX4]  6ØØ:            DiskSize(Intern + 1),' Bytes');
[Ø82]  6Ø1:    GotoXY ( 4, 8);
[DP5]  6Ø2:    Write  ('Freie Disk-Kapazität     : ',
[ØV4]  6Ø3:            DiskFree(Intern + 1),' Bytes');
[282]  6Ø4:    GotoXY ( 4, 9);
[L36]  6Ø5:    Write  ('Belegte Disk-Kapazität   : ',
[MD3]  6Ø6:            DiskSize(Intern + 1)-
[ØV4]  6Ø7:            DiskFree(Intern + 1),' Bytes');
[F36]  6Ø8:    xWindow (' Disk-Laufwerke ',51,12,78,23);
[2G2]  6Ø9:    For e := 1 To 6 Do
[MUØ]  61Ø:    Begin
[7N2]  611:      GotoXY ( 3, e + 2);
[ØØØ]  612:
[T12]  613: {$IFDEF LW1}
[ØØØ]  614:
[FS1]  615:      If e = 2 Then
[AZ3]  616:        Write (Chr (e+64),': ---------')
[WP1]  617:      Else Begin
[ØØØ]  618:
[WD1]  619: {$ENDIF}
[ØØØ]  62Ø:
[BA3]  621:        Write (Chr (e + 64),': ');
[CS1]  622:        Reg.DL := e;
[1T1]  623:        Reg.AH := 28;
[8T1]  624:        MsDos (Reg);
[943]  625:        If (Reg.AL = 255) Then
[4E2]  626:          Write ('---------')
[ZRØ]  627:          Else
[MUØ]  628:        Begin
[2W3]  629:          Bytes := Reg.AL * Reg.CX;
[EW3]  63Ø:          Bytes := Bytes * Reg.DX;
[PK3]  631:          Write (Bytes,' Bytes');
[ZNØ]  632:        End;
[ØØØ]  633:
[T12]  634: {$IFDEF LW1}
[ØØØ]  635:
[ZNØ]  636:      End;
[ØØØ]  637:
[WD1]  638: {$ENDIF}
[ØØØ]  639:
[ZNØ]  64Ø:    End;
[3B3]  641:    xWindow ('',3,2Ø,48,23);
[WP5]  642:   { DOS-Versionsnummer ausgeben }
[Ø82]  643:    GotoXY ( 8, 2);
[9T1]  644:    Reg.AH := 48;
[8T1]  645:    MsDos (Reg);
[AX5]  646:    Write ('Installierte Dos-Version ',
[WS2]  647:           Reg.AL,'.',Reg.AH);
[2Z1]  648:    c := ReadKey;
[ZTØ]  649:    Ende;
[ZNØ]  65Ø: End;
[ØØØ]  651:
[KNØ]  652: End.
```

Listing 2. In der Unit »general.pas« sind allgemeine Programmroutinen enthalten

```
[324]    1: { Programmname: DISK.PAS     }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[6A2]    5: {$DEFINE LW1}
[WK5]    6: { Zeile weglassen, wenn mit zwei
[5W4]    7:   Floppys gearbeitet wird !      }
[ØØØ]    8:
[LE2]    9: {  U N I T   D I S K
[X24]   1Ø:    Autor: Karlheinz Büker
[ØØØ]   11:
[ZE5]   12:   Enthält Prozeduren zu diversen
[455]   13:   Directory- und Dateifunktionen
[F65]   14:   von EASYDOS, sowie zugehörige
[NW3]   15:   Subroutinen                   }
[ØØØ]   16:
[PP1]   17: Unit Disk;
[ØØØ]   18:
[SK1]   19: INTERFACE
[ØØØ]   2Ø:
[ASØ]   21: Uses
```

Listing 3. Die Unit »disk.pas« stellt Datei- und Verzeichnis-Funktionen zur Verfügung

```
[L74]   22:   Dos, Crt, Printer, Global;
[ØØØ]   23:
[T12]   24: {$IFDEF LW1}
[6XØ]   25: Const
[SW1]   26:   LW2 = False;
[WD1]   27: {$ENDIF}
[ØØØ]   28:
[T92]   29: {$IFNDEF LW1}
[6XØ]   3Ø: Const
[6V1]   31:   LW2 = True;
[WD1]   32: {$ENDIF}
[ØØØ]   33:
[STØ]   34: Type
[GT2]   35:   Str2 = String[2];
[ØØØ]   36:
[RNØ]   37: Var
[ZM1]   38:   c      : Char;
[1P2]   39:   Fehler : Boolean;
[ØØØ]   4Ø:
[Ø73]   41: Procedure ShowDir;
[P54]   42: { Directory anzeigen     }
[YH3]   43: Procedure SearchDir;
[XL4]   44: { Directory durchsuchen }
[2G3]   45: Procedure ChangeDir;
[S44]   46: { Directory wechseln     }
[343]   47: Procedure MakeDir;
[A34]   48: { Directory anlegen      }
[WH3]   49: Procedure ReMakeDir;
[XB4]   5Ø: { Directory löschen      }
[U64]   51: Procedure MakeHiddenDir;
[VZ5]   52: { verstecktes Directory anlegen }
[G73]   53: Procedure SetAttr;
[TL4]   54: { Attribute setzen               }
[DD3]   55: Procedure MoveFile;
[YW5]   56: { Datei umhängen in anderes Verz}
[8X3]   57: Procedure SetFileTime;
[CC5]   58: { Datei-Zeit und Datum ändern    }
[KN3]   59: Procedure RenameFile;
[ZF4]   6Ø: { Datei umbenennen               }
[YM3]   61: Procedure DeleteFile;
[5G4]   62: { Datei löschen                  }
[ND3]   63: Procedure CopyFile;
[AA4]   64: { Datei kopieren                 }
[6F3]   65: Procedure ShowFile;
[Y94]   66: { Datei anzeigen                 }
[4L3]   67: Procedure PrintFile;
[E94]   68: { Datei drucken                  }
[TG3]   69: Procedure ShowTree;
[4X5]   7Ø: { Verzeichnisstruktur anzeigen   }
[AN3]   71: Procedure SearchFile;
[XN4]   72: { Dateieintrag suchen            }
[1N3]   73: Procedure DosCommand;
[CE5]   74: { DOS-Kommandozeile eingeben     }
[YL4]   75: Procedure BLW ( Test : Str2 );
[WN4]   76: { B-Laufwerk testen              }
[ØØØ]   77:
[4L2]   78: IMPLEMENTATION
[ØØØ]   79:
[STØ]   8Ø: Type
[2F2]   81:   Index       = 1..1ØØ;
[K54]   82:   PathString  = String[64];
[194]   83:   DateiString = String[12];
[A53]   84:   DirSatz     = Record
[E33]   85:                   Name     : String[12];
[N53]   86:                   Ext      : String[4];
[S72]   87:                   Attr     : Byte;
[AP2]   88:                   Zeit     : LongInt;
[733]   89:                   Groesse  : LongInt;
[ZNØ]   9Ø:                 End;
[ØØØ]   91:
[RNØ]   92: Var
[7M1]   93:   Position,
[ED1]   94:   Zaehler,
[EK1]   95:   Satzzahl,
[E11]   96:   First,
[UE1]   97:   Auswahl,
[1Ø1]   98:   Zeile,
[N72]   99:   Durchsuchen,
[CF1]  1ØØ:   Gefunden,
[UF2]  1Ø1:   Verzeichnisse,
[FT2]  1Ø2:   a                  : Byte;
[FHØ]  1Ø3:   OK,
[134]  1Ø4:   Dateien            : Boolean;
[LUØ]  1Ø5:   Zeit,
[DV3]  1Ø6:   Files              : LongInt;
[G9Ø]  1Ø7:   v,
[4Y2]  1Ø8:   LW                 : Char;
[ZY3]  1Ø9:   SRec               : SearchRec;
[RP4]  11Ø:   AlterPfad          : String[64];
[T33]  111:   Test               : Str2;
[HS4]  112:   Speicher           : Array [Ø..255]
[9B2]  113:                        of String[64];
[ØØØ]  114:
[SY4]  115: { Array zur Speicherung der
[TF4]  116:   Unterverzeichnisse       }
[4M3]  117:   DT                 : DateTime;
[991]  118:   SubDir,
[KSØ]  119:   Pfad,
[W64]  12Ø:   Pfad2              : PathString;
```

```
[MY4]  121:   Dateiname          : DateiString;
[GM4]  122:   Satz               : Array [Index]
[BZ1]  123:                        of DirSatz;
[621]  124:   Horiz,
[SZ3]  125:   Vert               : ShortInt;
[ØØØ]  126:
[3B5]  127: {--- Prüft B-Laufwerk (Steuerung
[WW4]  128:      über $DEFINE) ----------------------}
[ØØØ]  129:
[YL4]  13Ø: Procedure BLW ( Test : Str2 );
[MUØ]  131: Begin
[MG2]  132:   Fehler := False;
[XF5]  133:   If (Test = 'B:') Or (Test = 'b:') Then
[MUØ]  134:   Begin
[LV2]  135:     If LW2 = False Then
[MUØ]  136:     Begin
[LV1]  137:       CursorOff;
[2F2]  138:       Fehler := True;
[Y23]  139:       Fehlermeldung (19);
[D23]  14Ø:       Fehlermeldung (2Ø);
[MN1]  141:       CursorOn;
[ZNØ]  142:     End;
[ZNØ]  143:   End;
[ZNØ]  144: End;
[ØØØ]  145:
[DM6]  146: {--- Speicher-Array mit Verzeichnissen
[YY5]  147:      und Sub-Verzeichnissen laden -------}
[ØØØ]  148:
[9U2]  149: Procedure Suche;
[MUØ]  15Ø: Begin
[P31]  151:   Repeat
[ZW5]  152:     FindFirst (Speicher[Durchsuchen] +
[NP3]  153:                '\*.*',Directory,SRec);
[H63]  154:     While DosError = Ø Do
[MUØ]  155:     Begin
[BE4]  156:       If ((SRec.Attr And $1Ø) = 16) And
[WØ3]  157:          (SRec.Name <> '.') And
[1D3]  158:          (Srec.Name <> '..') Then
[MUØ]  159:       Begin
[FC3]  16Ø:         Inc (Verzeichnisse);
[7A2]  161:         Inc(Gefunden);
[CL3]  162:         Speicher[Gefunden] :=
[434]  163:           Speicher[Durchsuchen] +
[3C2]  164:           '\' + SRec.Name;
[ZNØ]  165:       End;
[5P2]  166:       FindNext(SRec);
[ZNØ]  167:     End;
[C13]  168:     Inc (Durchsuchen);
[GZ5]  169:   Until (Speicher[Durchsuchen] = '') Or
[NØ3]  17Ø:         (Durchsuchen > 255);
[ZNØ]  171: End;
[ØØØ]  172:
[DY4]  173: {--- Fenster löschen und auf
[GE5]  174:      Tastendruck warten -----------------}
[ØØØ]  175:
[RH3]  176: Procedure Blaettern;
[MUØ]  177: Begin
[P31]  178:   Repeat
[2Z1]  179:     c := Readkey;
[912]  18Ø:   Until c <> '';
[DB1]  181:   ClrScr;
[CH1]  182:   Vert := 1;
[ZNØ]  183: End;
[ØØØ]  184:
[ZM5]  185: {--- Wechsel von Bildschirmseiten
[X65]  186:      nach Tastendruck -------------------}
[ØØØ]  187:
[9V2]  188: Procedure Seite;
[MUØ]  189: Begin
[NYØ]  19Ø:   v := #Ø;
[P31]  191:   Repeat
[8Ø2]  192:     v := ReadKey;
[B22]  193:   Until v = #13;
[1P1]  194:   Zeile := 1;
[DB1]  195:   ClrScr;
[ZNØ]  196: End;
[ØØØ]  197:
[ØM4]  198: {--- Anzeige von gefundenen
[8H5]  199:      (Sub-)Verzeichnissen ---------------}
[ØØØ]  2Ø0:
[A55]  2Ø1: Procedure VerzeichnisAusgabe;
[MUØ]  2Ø2: Begin
[1P1]  2Ø3:   Zeile := 1;
[C82]  2Ø4:   GotoXY (2Ø, 2);
[ØW6]  2Ø5:   WriteLn (Verzeichnisse,' Verzeichnisse'+
[775]  2Ø6:            ' / Subverzeichnisse gefunden');
[XF1]  2Ø7:   Writeln;
[F52]  2Ø8:   Inc (Zeile,2);
[284]  2Ø9:   For a := Ø To Durchsuchen Do
[MUØ]  21Ø:   Begin
[9W3]  211:     Writeln (Speicher[a]);
[9X1]  212:     Inc (Zeile);
[NP3]  213:     If Zeile > 21 Then Seite;
[ZNØ]  214:   End;
[ZNØ]  215: End;
[ØØØ]  216:
[MK5]  217: {--- Anzeige von gefundenen (Sub-)
[LD6]  218:      Verzeichnissen und Dateieinträgen --}
[ØØØ]  219:
```

```
[734]  220: Procedure DateiAusgabe;
[MU0]  221: Begin
[UT2]  222:   Durchsuchen := 0;
[1P1]  223:   Zeile := 1;
[P31]  224:   Repeat
[G84]  225:     TextColors (Black,White);
[K84]  226:     Writeln ('Verzeichnis : ',
[Z54]  227:              Speicher[Durchsuchen]);
[S64]  228:     TextColors (White,Black);
[9X1]  229:     Inc (Zeile);
[NP3]  230:     If Zeile > 21 Then Seite;
[ZW5]  231:     FindFirst (Speicher[Durchsuchen] +
[NP3]  232:                '\*.*',Directory,SRec);
[H63]  233:     While DosError = 0 Do
[MU0]  234:     Begin
[N14]  235:       If ((SRec.Attr And $10) <> 16)
[T95]  236:         And ((SRec.Attr And $08) <> 8) Then
[MU0]  237:       Begin
[WW1]  238:         Inc (Files);
[1X3]  239:         WriteLn ('              -------> ',
[VV1]  240:                  SRec.Name);
[9X1]  241:         Inc (Zeile);
[NP3]  242:         If Zeile > 21 Then Seite;
[ZN0]  243:       End;
[AM2]  244:       FindNext (SRec);
[ZN0]  245:     End;
[GX4]  246:     Write (' ':22,'< KEINE WEITEREN '+
[5U3]  247:            'DATEIEINTRÄGE >',' ':23);
[9X1]  248:     Inc (Zeile);
[NP3]  249:     If Zeile > 21 Then Seite;
[C13]  250:     Inc (Durchsuchen);
[9V5]  251:   Until (Speicher[Durchsuchen] = '');
[811]  252:   Seite;
[DB1]  253:   ClrScr;
[S72]  254:   GotoXY ( 1,10);
[HK5]  255:   WriteLn ('           ',Verzeichnisse:3,
[H43]  256:          ' Verzeichnisse / '+
[3U4]  257:          'Subverzeichnisse gelesen ');
[CQ4]  258:   WriteLn ('           ',Files:3,
[BL4]  259:          ' Dateieinträge gefunden');
[ZN0]  260: End;
[000]  261:
[0M4]  262: {--- Anzeige von gefundenen
[ND5]  263:      Dateieinträgen ---------------------}
[000]  264:
[KN6]  265: Procedure GefundenAnzeigen (Dateiname :
[4A2]  266:                             DateiString);
[MU0]  267: Begin
[UT2]  268:   Durchsuchen := 0;
[P31]  269:   Repeat
[ZW5]  270:     FindFirst (Speicher[Durchsuchen] +
[NP3]  271:                '\*.*',Directory,SRec);
[H63]  272:     While DosError = 0 Do
[MU0]  273:     Begin
[HX4]  274:       If (SRec.Name = Dateiname) Then
[MU0]  275:       Begin
[E05]  276:         Writeln ('Eintrag gefunden : ',
[WY3]  277:                  Speicher[Durchsuchen],
[8E2]  278:                  '\',Dateiname);
[ZN0]  279:       End;
[AM2]  280:       FindNext (SRec);
[ZN0]  281:     End;
[C13]  282:     Inc (Durchsuchen);
[9V5]  283:   Until (Speicher[Durchsuchen] = '');
[XF1]  284:   Writeln;
[HK5]  285:   WriteLn ('           ',Verzeichnisse:3,
[H43]  286:          ' Verzeichnisse / '+
[3U4]  287:          'Subverzeichnisse gelesen ');
[ZN0]  288: End;
[000]  289:
[615]  290: {--- Warnung für versteckte
[VL5]  291:      Directories ausgeben ---------------}
[000]  292:
[783]  293: Procedure Warnung;
[MU0]  294: Begin
[RH4]  295:   xWindow ('  W  A  R  N  U  N  G  ',
[9D1]  296:           3,3,77,16);
[HB6]  297:   WriteXY (2,3,' Versteckte Directories '+
[G65]  298:           'werden von EASYDOS genauso '+
[8V3]  299:           'wie normale Directo-');
[WY5]  300:   WriteXY (2,5,' ries behandelt. Im MS-'+
[1R4]  301:           'DOS-Modus hingegen werden '+
[9X3]  302:           'diese nicht mehr ange-');
[4L5]  303:   WriteXY (2,7,' zeigt bzw. sind nur '+
[8F4]  304:           'noch mit Tricks sichtbar '+
[FE4]  305:           'zu machen. Bitte merken');
[R75]  306:   WriteXY (2,9,' Sie sich daher in '+
[0D4]  307:           'jedem Fall Namen und Pfad '+
[145]  308:           'versteckter Verzeichnisse.');
[ZN0]  309: End;
[000]  310:
[5A6]  311: {--- Ausgewählte Directory-Position
[L35]  312:      invertieren ------------------------}
[000]  313:
[1F3]  314: Procedure BalkenNeg;
[MU0]  315: Begin
[MB5]  316:   TextColors (White + Blink,Black);
[Z47]  317:   WriteXY (Horiz,Vert,Satz[Auswahl].Name);
[G84]  318:   TextColors (Black,White);
[ZN0]  319: End;
[000]  320:
[LH3]  321: Procedure BalkenPos;
[MU0]  322: Begin
[Z47]  323:   WriteXY (Horiz,Vert,Satz[Auswahl].Name);
[ZN0]  324: End;
[000]  325:
[ZA5]  326: {--- Steuerung des invertierten
[AR4]  327:      Balkens ----------------------------}
[000]  328:
[6L3]  329: Procedure BalkenHoch;
[MU0]  330: Begin
[9V3]  331:   If (Auswahl - 4) >= 1 Then
[MU0]  332:   Begin
[WS1]  333:     BalkenPos;
[DF2]  334:     Dec (Auswahl,4);
[AU1]  335:     Dec (Vert);
[KR1]  336:     BalkenNeg;
[ZN0]  337:   End;
[ZN0]  338: End;
[000]  339:
[KL3]  340: Procedure BalkenTief;
[MU0]  341: Begin
[465]  342:   If (Auswahl + 4) <= Satzzahl Then
[MU0]  343:   Begin
[WS1]  344:     BalkenPos;
[KG2]  345:     Inc (Auswahl,4);
[FV1]  346:     Inc (Vert);
[KR1]  347:     BalkenNeg;
[ZN0]  348:   End;
[ZN0]  349: End;
[000]  350:
[L34]  351: Procedure BalkenRechts;
[MU0]  352: Begin
[S55]  353:   If (Auswahl + 1) <= Satzzahl Then
[MU0]  354:   Begin
[WS1]  355:     BalkenPos;
[2A2]  356:     Inc (Auswahl);
[GD2]  357:     Inc (Horiz,15);
[PZ2]  358:     If (Horiz > 46) Then
[MU0]  359:     Begin
[6S1]  360:       Horiz := 1;
[FV1]  361:       Inc (Vert);
[ZN0]  362:     End;
[KR1]  363:     BalkenNeg;
[ZN0]  364:   End;
[ZN0]  365: End;
[000]  366:
[CV3]  367: Procedure BalkenLinks;
[MU0]  368: Begin
[8L3]  369:   If (Auswahl - 1) > 0 Then
[MU0]  370:   Begin
[WS1]  371:     BalkenPos;
[X82]  372:     Dec (Auswahl);
[BC2]  373:     Dec (Horiz,15);
[6W2]  374:     If (Horiz < 1) Then
[MU0]  375:     Begin
[PT1]  376:       Horiz := 46;
[AU1]  377:       Dec (Vert);
[ZN0]  378:     End;
[KR1]  379:     BalkenNeg;
[ZN0]  380:   End;
[ZN0]  381: End;
[000]  382:
[US6]  383: {-- Directory-Array sortieren -----------}
[000]  384:
[D45]  385: Procedure DirSort (B, E : Index;
[552]  386:                     Param : Char);
[RN0]  387: Var
[M90]  388:   x,
[2A2]  389:   Buf : DirSatz;
[680]  390:   a,
[EP1]  391:   k   : Index;
[MU0]  392: Begin
[TZ0]  393:   a := B;
[P01]  394:   k := E;
[5D3]  395:   x := Satz[(B + E) Div 2];
[P31]  396:   Repeat
[412]  397:     Case Param Of
[GE1]  398:       'E': Begin
[5N4]  399:              While Satz[a].Ext < x.Ext Do
[SV1]  400:                a := Succ(a);
[TU4]  401:              While x.Ext < Satz[k].Ext Do
[3X1]  402:                k := Pred(k);
[ZN0]  403:            End;
[BF1]  404:       'N': Begin
[K05]  405:              While Satz[a].Name < x.Name Do
[SV1]  406:                a := Succ(a);
[L05]  407:              While x.Name < Satz[k].Name Do
[3X1]  408:                k := Pred(k);
[ZN0]  409:            End;
[UF1]  410:       'S': Begin
[834]  411:              While Satz[a].Groesse <
[XZ1]  412:                    x.Groesse Do
[SV1]  413:                a := Succ(a);
[YX2]  414:              While x.Groesse <
```

Listing 3. Die Unit »disk.pas« stellt Datei- und Verzeichnis-Funktionen zur Verfügung

```
[TC3]  415:                    Satz[k].Groesse Do
[3X1]  416:                  k := Pred(k);
[ZNØ]  417:              End;
[4E1]  418:       'A': Begin
[845]  419:              While Satz[a].Attr < x.Attr Do
[SV1]  42Ø:                a := Succ(a);
[X55]  421:              While x.Attr < Satz[k].Attr Do
[3X1]  422:                k := Pred(k);
[ZNØ]  423:              End;
[XF1]  424:       'T': Begin
[F55]  425:              While Satz[a].Zeit < x.Zeit Do
[SV1]  426:                a := Succ(a);
[U35]  427:              While x.Zeit < Satz[k].Zeit Do
[3X1]  428:                k := Pred(k);
[ZNØ]  429:              End;
[ZNØ]  43Ø:     End;
[F12]  431:     if a <= k Then
[MUØ]  432:     Begin
[ZL2]  433:       Buf := Satz[a];
[SF3]  434:       Satz[a] := Satz[k];
[LM2]  435:       Satz[k] := Buf;
[SV1]  436:       a := Succ(a);
[3X1]  437:       k := Pred(k);
[ZNØ]  438:     End;
[CW1]  439:   Until a > k;
[F15]  44Ø:   If B < k Then DirSort(B,k,Param);
[SØ5]  441:   If a < E Then DirSort(a,E,Param);
[ZNØ]  442: End;
[ØØØ]  443:
[YU5]  444: {--- Masken für Directory-Anzeige
[8S4]  445:      vorgeben ---------------------------}
[ØØØ]  446:
[EU5]  447: Procedure Mask (Path : PathString);
[MUØ]  448: Begin
[MR2]  449:   Window (2,2,79,23);
[Y94]  45Ø:   TextColors (Yellow,Blue);
[B62]  451:   GotoXY  ( 2, 1);
[P32]  452:   Write (' ':77);
[B62]  453:   GotoXY  ( 2, 1);
[U95]  454:   Write   ('Directory von : ',Path);
[E62]  455:   GotoXY  ( 2, 2);
[DØ6]  456:   WriteXY ( 2, 3,'Name      Ext ▌ Bytes  '+
[XN5]  457:             '▌ Datum       ▌ Zeit  ▌ Attri'+
[V13]  458:             'bute                        ');
[YKD]  459:   WriteXY ( 2, 4,'═══════════╪═══════════'+
[WWD]  46Ø:             '╪═══════════╪═══════════'+
[GDE]  461:             '═══════════╧═══════════');
[XM4]  462:   WriteXY (68, 6,'ATTRIBUTE  ');
[AA8]  463:   WriteXY (68, 7,'──────────');
[RA4]  464:   WriteXY (68, 8,'H= Hidden  ');
[TP4]  465:   WriteXY (68, 9,'R= ReadOnly');
[DG4]  466:   WriteXY (68,1Ø,'S= System  ');
[HF4]  467:   WriteXY (68,11,'A= Archive ');
[YN4]  468:   WriteXY (68,15,' zum Menü  ');
[494]  469:   WriteXY (68,16,' --->HOME  ');
[AR4]  47Ø:   WriteXY (68,18,'  Blättern ');
[5M4]  471:   WriteXY (68,19,' PgUp/PgDn ');
[C72]  472:   GotoXY  ( 2,2Ø);
[2R6]  473:   Write (Satzzahl-1,' Dateien/Directorys '+
[FN3]  474:          'gefunden ',DiskFree(Ø),
[VP3]  475:          ' Bytes frei auf Disk');
[UR5]  476:   WriteXY ( 2,22,'SORT : ALT+N = Name / '+
[M94]  477:             'E = Extension / T = Zeit / '+
[YM4]  478:             'A = Attribute / S = Größe ');
[Ø34]  479:    DirSort (1,Satzzahl,'S');
[ZNØ]  48Ø: End;
[ØØØ]  481:
[DZ5]  482: Procedure Mask2 (Path : PathString);
[MUØ]  483: Begin
[MR2]  484:   Window (2,2,79,23);
[Y94]  485:   TextColors (Yellow,Blue);
[972]  486:   GotoXY ( 1, 1);
[S32]  487:   Write (' ':78);
[972]  488:   GotoXY ( 1, 1);
[W15]  489:   Write ('Directory von : ',Path);
[8G4]  49Ø:   WriteXY ( 1, 2,'                    D I '+
[V44]  491:             'R E C T O R Y     E I N T R Ä'+
[8U2]  492:             ' G E                        ');
[MKD]  493:   WriteXY ( 1, 3,'═══════════════════════'+
[XBE]  494:             '═══════════════════════'+
[GDE]  495:             '═══════════════════════');
[5K4]  496:   WriteXY (65,12,'    ÖFFNEN   ');
[E84]  497:   WriteXY (65,13,'    ---> CR  ');
[PE4]  498:   WriteXY (65,15,'   ABBRUCH   ');
[D84]  499:   WriteXY (65,16,'    --->ESC  ');
[285]  5ØØ:   WriteXY (65,18,' WÄHLEN MIT  ');
[M15]  5Ø1:   WriteXY (65,19,' PFEILTASTEN ');
[ZNØ]  5Ø2: End;
[ØØØ]  5Ø3:
[ZK4]  5Ø4: {--- Anzeigen der Directory-
[EY4]  5Ø5:      Einträge ---------------------------}
[ØØØ]  5Ø6:
[3M5]  5Ø7: Procedure DirOutput (F : DirSatz;
[812]  5Ø8:                      Vert : Byte);
[MUØ]  5Ø9: Begin
[8P2]  51Ø:   GotoXY ( 2,Vert);
[ZW3]  511:   If ((F.Attr And $Ø8) = 8) Or
[XX3]  512:      ((F.Attr And $1Ø) = 16) Then
[234]  513:     TextColors (White,Black)
[S45]  514:   Else TextColors (Black,White);
[MB2]  515:   Write (F.Name);
[CX3]  516:   If (F.Attr And $1Ø) = 16 Then
[364]  517:     WriteXY (1Ø,Vert,'<DIR>');
[BØ4]  518:   WriteXY (15,Vert,'▌');
[UV3]  519:   UnpackTime (F.Zeit,DT);
[KS1]  52Ø:   With DT Do
[MUØ]  521:   Begin
[WR2]  522:     GotoXY (18,Vert);
[H14]  523:     Write  (F.Groesse:6,'▌');
[SR2]  524:     GotoXY (26,Vert);
[575]  525:     Write  (Day:2,'.',Month:2,'.',Year,
[8P4]  526:             ' ▌ ',Hour:2,':',Min:2,' ▌');
[ZNØ]  527:   End;
[ZR2]  528:   GotoXY (47,Vert);
[4V3]  529:   If (F.Attr And $Ø1) =  1 Then
[X22]  53Ø:     Write ('<R>');
[8V3]  531:   If (F.Attr And $Ø2) =  2 Then
[Z12]  532:     Write ('<H>');
[GV3]  533:   If (F.Attr And $Ø4) =  4 Then
[Ø32]  534:     Write ('<S>');
[ØW3]  535:   If (F.Attr And $Ø8) =  8 Then
[7M3]  536:     Write ('<DISK - LABEL>');
[CX3]  537:   If (F.Attr And $1Ø) = 16 Then
[AK3]  538:     Write ('<DIRECTORY>');
[KX3]  539:   If (F.Attr And $2Ø) = 32 Then
[A12]  54Ø:     Write ('<A>');
[G84]  541:   TextColors (Black,White);
[ZNØ]  542: End;
[ØØØ]  543:
[GM6]  544: {--- Directory in Array einlesen --------}
[ØØØ]  545:
[EU6]  546: Procedure DateiEinlesen (Searchstring :
[Y32]  547:                          Pathstring);
[RNØ]  548: Var
[EM2]  549:   Free  : LongInt;
[C62]  55Ø:   Punkt : Byte;
[MUØ]  551: Begin
[MG2]  552:   Fehler := False;
[982]  553:   Satzzahl := Ø;
[9L6]  554:   FindFirst (Searchstring,AnyFile,SRec);
[CM3]  555:   If (DosError <> Ø) Then
[MUØ]  556:   Begin
[FT2]  557:     Fehlermeldung(1);
[DB1]  558:     ClrScr;
[2F2]  559:     Fehler := True;
[ZNØ]  56Ø:   End;
[C73]  561:   If Fehler Then Exit;
[H33]  562:   Free := DiskFree(Ø);
[WR3]  563:   While (DosError = Ø) And
[YY2]  564:         (Satzzahl < 1Ø1) Do
[MUØ]  565:   Begin
[PK2]  566:     Inc (Satzzahl);
[LL5]  567:     Satz[Satzzahl].Name := SRec.Name;
[2G2]  568:     For a := 1 To 8 Do
[MUØ]  569:     Begin
[936]  57Ø:       If (Satz[Satzzahl].Name[a] = '.') Then
[UT5]  571:         Insert (' ',Satz[Satzzahl].Name,a);
[ZNØ]  572:     End;
[NB4]  573:     Punkt := Pos ('.',SRec.Name);
[ØE6]  574:     Satz[Satzzahl].Ext  := Copy(SRec.Name,
[XD1]  575:                                 Punkt,4);
[XS5]  576:     Satz[Satzzahl].Attr := SRec.Attr;
[VP5]  577:     Satz[Satzzahl].Zeit := SRec.Time;
[266]  578:     Satz[Satzzahl].Groesse := SRec.Size;
[AM2]  579:     FindNext (SRec);
[ZNØ]  58Ø:   End;
[ZNØ]  581: End;
[ØØØ]  582:
[8Z4]  583: {--- Directory-Inhalt ausgeben
[FT5]  584:      und einen Satz auswählen -----------}
[ØØØ]  585:
[RL3]  586: Procedure WaehleAus;
[MUØ]  587: Begin
[8A2]  588:   Dateiname := '';
[Z34]  589:   TextColors (Blue,White);
[TR2]  59Ø:   Window (2,2,79,16);
[DB1]  591:   ClrScr;
[6S1]  592:   Horiz := 1;
[VM1]  593:   Vert  := 1;
[FU3]  594:   For a := 1 to Satzzahl Do
[MUØ]  595:   Begin
[L46]  596:     WriteXY (Horiz,Vert,Satz[a].Name);
[KD2]  597:     Inc (Horiz,16);
[RZ2]  598:     If (Horiz > 65) Then
[MUØ]  599:     Begin
[6S1]  6ØØ:       Horiz := 1;
[FV1]  6Ø1:       Inc (Vert);
[ZNØ]  6Ø2:     End;
[ZNØ]  6Ø3:   End;
[MN1]  6Ø4:   CursorOn;
[S64]  6Ø5:   TextColors (White,Black);
[5L6]  6Ø6:   XWindow (' Bitte Dateinamen aus aktue'+
[825]  6Ø7:            'llem Directory auswählen : ',
[3E1]  6Ø8:           3,2Ø,77,23);
[P31]  6Ø9:   Repeat
[H72]  61Ø:     GotoXY ( 3, 2);
```

```
[343]  611:     ReadLn (DateiName);
[ME3]  612:   Until Dateiname <> '';
[DG4]  613:   Test := Copy (Dateiname,1,2);
[6X1]  614:   BLW (Test);
[H82]  615:   If Fehler Then
[MUØ]  616:   Begin
[ZTØ]  617:     Ende;
[ØWØ]  618:     Exit;
[ZNØ]  619:   End;
[LV1]  62Ø:   CursorOff;
[PC3]  621:   Assign (f,Dateiname);
[DXØ]  622:   {$I-}
[NF1]  623:     Reset (f);
[BXØ]  624:   {$I+}
[5A3]  625:   If IOResult <> Ø Then
[MUØ]  626:   Begin
[CW2]  627:     Fehlermeldung (5);
[ØWØ]  628:     Exit;
[ZNØ]  629:   End;
[CE1]  63Ø:   Close (f);
[ZNØ]  631: End;
[ØØØ]  632:
[BX6]  633: {--- Steuerung der Directory-Ausgabe ----}
[ØØØ]  634:
[PL3]  635: Procedure DirAusgabe;
[ØUØ]  636: Label
[Y72]  637:   SortOutput;
[MUØ]  638: Begin
[TR2]  639:   Window (2,6,68,2Ø);
[U72]  64Ø: SortOutput:
[G84]  641:   TextColors (Black,White);
[MUØ]  642:   Begin
[DB1]  643:     ClrScr;
[CU1]  644:     OK := False;
[VE3]  645:     If Satzzahl >= 15 Then
[MUØ]  646:     Begin
[EP2]  647:       For a := 1 to 15 Do
[6T3]  648:         DirOutput(Satz[a],a);
[T42]  649:       Zaehler := 15;
[4MØ]  65Ø:     End
[ZRØ]  651:     Else
[MUØ]  652:     Begin
[FU3]  653:       For a := 1 to Satzzahl Do
[6T3]  654:         DirOutput(Satz[a],a);
[5D3]  655:       Zaehler := Satzzahl;
[ZNØ]  656:     End;
[2Z1]  657:     c := ReadKey;
[DW1]  658:     If c = #Ø Then
[P31]  659:     Repeat
[2Z1]  66Ø:       c := ReadKey;
[Ø72]  661:       Case Ord(c) Of
[333]  662:        31 : Begin { ALT - S }
[114]  663:               DirSort(1,Satzzahl,'S');
[823]  664:               Goto SortOutput;
[ZNØ]  665:             End;
[633]  666:        18 : Begin { ALT - E }
[5Ø4]  667:               DirSort(1,Satzzahl,'E');
[823]  668:               Goto SortOutput;
[ZNØ]  669:             End;
[ØØØ]  67Ø:
[S33]  671:        49 : Begin { ALT - N }
[RØ4]  672:               DirSort(1,Satzzahl,'N');
[823]  673:               Goto SortOutput;
[ZNØ]  674:             End;
[Z23]  675:        2Ø : Begin { ALT - T }
[314]  676:               DirSort(1,Satzzahl,'T');
[823]  677:               Goto SortOutput;
[ZNØ]  678:             End;
[E23]  679:        3Ø : Begin { ALT - A }
[XZ3]  68Ø:               DirSort(1,Satzzahl,'A');
[823]  681:               Goto SortOutput;
[ZNØ]  682:             End;
[E13]  683:        73 : Begin { PgUp }
[2Z2]  684:               If Zaehler > 15 Then
[MUØ]  685:               Begin
[DB1]  686:                 ClrScr;
[LL3]  687:                 Zaehler := Zaehler - 15;
[W64]  688:                 If Zaehler > Satzzahl Then
[5D3]  689:                   Zaehler := Satzzahl;
[ØZ2]  69Ø:                 If Zaehler < 15 Then
[T42]  691:                   Zaehler := 15;
[1D3]  692:                 For a:=Zaehler - 14 To
[RS1]  693:                         Zaehler Do
[3C3]  694:                          DirOutput(Satz[a],
[GE2]  695:                          a + 15 - Zaehler);
[ZNØ]  696:               End;
[ZNØ]  697:             End;
[NØ3]  698:        81 : Begin { PgDn }
[1E3]  699:               If (Zaehler <= 1Ø1) And
[633]  7ØØ:                  (Satzzahl > 15) Then
[MUØ]  7Ø1:               Begin
[DB1]  7Ø2:                 ClrScr;
[HL3]  7Ø3:                 Zaehler := Zaehler + 15;
[T13]  7Ø4:                 If Zaehler > 1ØØ Then
[UA2]  7Ø5:                   Zaehler := 1ØØ;
[W64]  7Ø6:                 If Zaehler > Satzzahl Then
[5D3]  7Ø7:                   Zaehler := Satzzahl;
[1D3]  7Ø8:                 For a:=Zaehler - 14 To
[RS1]  7Ø9:                         Zaehler Do
[3C3]  71Ø:                          DirOutput(Satz[a],
[GE2]  711:                          a + 15 - Zaehler);
[ZNØ]  712:               End;
[ZNØ]  713:             End;
[A72]  714:        71 : OK := True
[PP2]  715:        Else OK := False;
[ZNØ]  716:       End;
[2H1]  717:     Until OK;
[ZNØ]  718:   End;
[ZNØ]  719: End;
[ØØØ]  72Ø:
[NS3]  721: Procedure DirAusgabe2;
[MUØ]  722: Begin
[KR2]  723:   Window (2,5,65,23);
[G84]  724:   TextColors (Black,White);
[DB1]  725:   ClrScr;
[6S1]  726:   Horiz := 1;
[VM1]  727:   Vert  := 1;
[FU3]  728:   For a := 1 to Satzzahl Do
[MUØ]  729:   Begin
[L46]  73Ø:     WriteXY (Horiz,Vert,Satz[a].Name);
[GD2]  731:     Inc (Horiz,15);
[PZ2]  732:     If (Horiz > 46) Then
[MUØ]  733:     Begin
[6S1]  734:       Horiz := 1;
[FV1]  735:       Inc (Vert);
[ZNØ]  736:     End;
[ZNØ]  737:   End;
[Ø12]  738:   Auswahl := 1;
[MS1]  739:   Horiz   := 1;
[9N1]  74Ø:   Vert    := 1;
[ZNØ]  741: End;
[ØØØ]  742:
[327]  743: {--- Steuerung des Directory-Cursors ----}
[ØØØ]  744:
[PF4]  745: Procedure AuswahlSteuern;
[MUØ]  746: Begin
[P31]  747:   Repeat
[KR1]  748:     BalkenNeg;
[P31]  749:     Repeat
[2Z1]  75Ø:       c := ReadKey;
[DW1]  751:       If c = #Ø Then
[MUØ]  752:       Begin
[2Z1]  753:         c := ReadKey;
[Ø72]  754:         Case Ord(c) Of
[VD2]  755:           72 : BalkenHoch;
[ND2]  756:           8Ø : BalkenTief;
[RL2]  757:           75 : BalkenLinks;
[GT2]  758:           77 : BalkenRechts;
[ZNØ]  759:         End;
[ZNØ]  76Ø:       End;
[ER3]  761:     Until (c = #13) Or (c = #27);
[H82]  762:     If (c = #13) Then
[MUØ]  763:     Begin
[ZB4]  764:       If (Satz[Auswahl].Attr And
[4U1]  765:           $1Ø) = 16 Tnen
[ØB5]  766:           { Nur wenn Auswahl = Directory }
[MUØ]  767:       Begin
[CN3]  768: { 1. Möglichkeit: cd .
[5U3]  769:   = Stammverzeichnis   }
[SZ3]  77Ø:         If Satz[Auswahl].Name =
[PR1]  771:              '       .' Then
[P84]  772: { 8 Leerstellen + Punkt }
[MUØ]  773:         Begin
[EC4]  774:           Position := Pos (':',Pfad);
[T93]  775:           If Position <> Ø Then
[MUØ]  776:           Begin
[NH4]  777:             For a := Position + 1 To 99 Do
[ZV2]  778:               Insert (' ',Pfad,a);
[1Z2]  779:             Insert ('\*.*',Pfad,
[452]  78Ø:                     Position + 1);
[ZNØ]  781:           End;
[ZNØ]  782:         End;
[8V3]  783: { 2. Möglichkeit: cd ..
[YL5]  784:   = Übergeordnetes Verzeichnis }
[SZ3]  785:         If Satz[Auswahl].Name =
[MV1]  786:              '       ..' Then
[SG4]  787: { 8 Leerstellen + 2 Punkte }
[MUØ]  788:         Begin
[MZØ]  789:           a := 64;
[2A2]  79Ø:           Position := Ø;
[BR1]  791:           First := Ø;
[X84]  792:           While (a > 1) And (First < 2) Do
[MUØ]  793:           Begin
[TE3]  794:             If (Pfad[a] = '\') Then
[MUØ]  795:             Begin
[4B2]  796:               Position := a;
[3Y2]  797:               First := First + 1;
[ZNØ]  798:             End;
[N31]  799:             Dec (a);
[ZNØ]  8ØØ:           End;
[S53]  8Ø1:           If Position > Ø Then
[MUØ]  8Ø2:           Begin
[BK4]  8Ø3:             Delete (Pfad,Position + 1,99);
```

Listing 3. Die Unit »disk.pas« stellt Datei- und Verzeichnis-Funktionen zur Verfügung

```
[WG9]  8Ø4:              Insert('*.*
                                                ',P
       fad,Position + 1);
[916]  8Ø5: { 1. Parameter = '*.*' + 61 Leerstellen }
[ZNØ]  8Ø6:            End;
[ZNØ]  8Ø7:          End;
[3K6]  8Ø8: { 3. Möglichkeit = Unterverzeichnis }
[U34]  8Ø9:          If (Satz[Auswahl].Name <>
[3P1]  81Ø:              '          ..') And
[UT3]  811:            (Satz[Auswahl].Name <>
[UU1]  812:              '          .') Then
[MUØ]  813:          Begin
[MZØ]  814:            a := 64;
[2A2]  815:            Position := Ø;
[Ø83]  816:            While Position = Ø Do
[MUØ]  817:            Begin
[D63]  818:              If Pfad[a] = '\' Then
[4B2]  819:                Position := a;
[N31]  82Ø:              Dec (a);
[ZNØ]  821:            End;
[SG4]  822:            For a := Position + 1 To 64 Do
[ZV2]  823:              Insert (' ',Pfad,a);
[RT4]  824:            Insert (Satz[Auswahl].Name+
[3M3]  825:                    '\*.*',Pfad,Position+1);
[ZNØ]  826:          End;
[ZNØ]  827:        End;
[NC3]  828:        DateiEinlesen (Pfad);
[A14]  829:        DirSort (1,Satzzahl,'A');
[6X1]  83Ø:        Mask2 (Pfad);
[222]  831:        DirAusgabe2;
[ZNØ]  832:      End;
[A42]  833:    Until (c = #27);
[ZNØ]  834: End;
[ØØØ]  835:
[W27]  836: {--- Hauptprogramm Directory Anzeigen ---}
[ØØØ]  837:
[Ø73]  838: Procedure ShowDir;
[MUØ]  839: Begin
[DB1]  84Ø:   ClrScr;
[G84]  841:   TextColors (Black,White);
[DB1]  842:   ClrScr;
[Ø25]  843:   xWindow (' D I R E C T O R Y   A N Z '+
[BT2]  844:            'E I G E N ',1,1,8Ø,25);
[WE1]  845:   AktPath;
[S64]  846:   TextColors (White,Black);
[MS2]  847:   Window (3,2Ø,78,22);
[DB1]  848:   ClrScr;
[3A6]  849:   xWindow (' Bitte Pfad und Dateimaske '+
[383]  85Ø:            'eingeben! ',3,2Ø,78,23);
[H72]  851:   GotoXY ( 3, 2);
[MN1]  852:   CursorOn;
[252]  853:   ReadLn (Pfad);
[LV1]  854:   CursorOff;
[ZK3]  855:   Test := Copy (Pfad,1,2);
[6X1]  856:   BLW (Test);
[H82]  857:   If Fehler Then
[MUØ]  858:   Begin
[ZTØ]  859:     Ende;
[ØWØ]  86Ø:     Exit;
[ZNØ]  861:   End;
[984]  862:   For a := 1 To Length(Pfad) Do
[BH4]  863:     Pfad[a] := Upcase(Pfad[a]);
[DC2]  864:   If Pfad = '' Then
[MUØ]  865:   Begin
[SD2]  866:     GetDir (Ø,Pfad);
[CP2]  867:     Pfad := Pfad+'*.*';
[ZNØ]  868:   End;
[E53]  869:   WriteXY ( 3, 2,Pfad);
[UR1]  87Ø:   Delay (1ØØØ);
[G84]  871:   TextColors (Black,White);
[DB1]  872:   ClrScr;
[NC3]  873:   DateiEinlesen (Pfad);
[H82]  874:   If Fehler Then
[MUØ]  875:   Begin
[ZTØ]  876:     Ende;
[ØWØ]  877:     Exit;
[ZNØ]  878:   End;
[ZU1]  879:   Mask (Pfad);
[B12]  88Ø:   DirAusgabe;
[ZTØ]  881:   Ende;
[ZNØ]  882: End;
[ØØØ]  883:
[LU4]  884: {--- Hauptprogramm Directory
[F35]  885:      durchsuchen ------------------------}
[ØØØ]  886:
[YH3]  887: Procedure SearchDir;
[MUØ]  888: Begin
[MG2]  889:   Fehler := False;
[DB1]  89Ø:   ClrScr;
[G84]  891:   TextColors (Black,White);
[DB1]  892:   ClrScr;
[175]  893:   xWindow (' D I R E C T O R Y   D U R '+
[8F3]  894:            'C H S U C H E N ',1,1,8Ø,25);
[WE1]  895:   AktPath;
[S64]  896:   TextColors (White,Black);
[MS2]  897:   Window (3,2Ø,78,22);
[DB1]  898:   ClrScr;
```

```
[5C6]  899:   xWindow (' Bitte Pfad für Suchbeginn '+
[2E4]  9ØØ:            'und Dateimaske eingeben! ',
[7E1]  9Ø1:            3,2Ø,78,23);
[H72]  9Ø2:   GotoXY ( 3, 2);
[MN1]  9Ø3:   CursorOn;
[252]  9Ø4:   ReadLn (Pfad);
[LV1]  9Ø5:   CursorOff;
[ZK3]  9Ø6:   Test := Copy (Pfad,1,2);
[6X1]  9Ø7:   BLW (Test);
[H82]  9Ø8:   If Fehler Then
[MUØ]  9Ø9:   Begin
[ZTØ]  91Ø:     Ende;
[ØWØ]  911:     Exit;
[ZNØ]  912:   End;
[984]  913:   For a := 1 to Length(Pfad) Do
[BH4]  914:     Pfad[a] := Upcase(Pfad[a]);
[DC2]  915:   If Pfad = '' Then
[MUØ]  916:   Begin
[SD2]  917:     GetDir (Ø,Pfad);
[1S2]  918:     Pfad := Pfad + '*.*';
[ZNØ]  919:   End;
[E53]  92Ø:   WriteXY ( 3, 2,Pfad);
[UR1]  921:   Delay (1ØØØ);
[G84]  922:   TextColors (Black,White);
[DB1]  923:   ClrScr;
[NC3]  924:   DateiEinlesen (Pfad);
[H82]  925:   If Fehler Then
[MUØ]  926:   Begin
[ZTØ]  927:     Ende;
[ØWØ]  928:     Exit;
[ZNØ]  929:   End;
[A14]  93Ø:   DirSort (1,Satzzahl,'A');
[6X1]  931:   Mask2 (Pfad);
[222]  932:   DirAusgabe2;
[BW2]  933:   AuswahlSteuern;
[ZTØ]  934:   Ende;
[ZNØ]  935: End;
[ØØØ]  936:
[647]  937: {--- Hauptprogramm Directory wechseln ---}
[ØØØ]  938:
[2G3]  939: Procedure ChangeDir;
[MUØ]  94Ø: Begin
[NR5]  941:   For a := Ø To 255 Do Speicher[a] := '';
[DB1]  942:   ClrScr;
[Ø35]  943:   xWindow (' S T A N D A R D D I R E C '+
[ØB3]  944:            'T O R Y   W E C H S E L N ',
[9C1]  945:            1,1,8Ø,25);
[WE1]  946:   AktPath;
[MR2]  947:   Window (2,2,79,23);
[W82]  948:   GotoXY (26, 1);
[VV4]  949:   Writeln ('Verzeichnisse im '+
[W93]  95Ø:            'Standardlaufwerk');
[2B3]  951:   GetDir (Ø,AlterPfad);
[ZE6]  952:  { aktuelles Verzeichnis speichern }
[C75]  953:   Speicher[Ø] := AlterPfad[1]+':';
[P57]  954:  { Startverzeichnis in SpeicherØ sichern }
[W63]  955:   ChDir (Speicher[Ø]);
[ZD6]  956:  { ins Stammverzeichnis wechseln          }
[UT2]  957:   Durchsuchen := Ø;
[UL6]  958:  { zu durchsuchendes Verzeichnis          }
[752]  959:   Gefunden := Ø;
[PW5]  96Ø:  { gefundene Verzeichnisse                }
[U53]  961:   Verzeichnisse := 1;
[7E6]  962:  { Anzahl gelesener Verzeichnisse         }
[3P1]  963:   Files := Ø;
[9Ø6]  964:  { Anzahl gefundener Dateien              }
[M11]  965:   Suche;
[3M6]  966:  { Directory-Array initialisieren         }
[TG3]  967:   VerzeichnisAusgabe;
[LU2]  968:   ChDir (AlterPfad);
[VX4]  969:  { wieder ins ursprüngliche
[9F5]  97Ø:    Verzeichnis zurück                     }
[PK6]  971:   xWindow (' Derzeit gesetztes Standard'+
[1U3]  972:            'directory : ',3,19,78,22);
[TE2]  973:   GetDir(Ø,Pfad);
[E53]  974:   WriteXY ( 3, 2,Pfad);
[MN1]  975:   CursorOn;
[VC6]  976:   XWindow (' Bitte neuen Pfad eingeben : ',
[FE1]  977:            3,22,78,25);
[EY1]  978:   GotoXY (3,2);
[252]  979:   ReadLn (Pfad);
[ZK3]  98Ø:   Test := Copy (Pfad,1,2);
[6X1]  981:   BLW (Test);
[H82]  982:   If Fehler Then
[MUØ]  983:   Begin
[ZTØ]  984:     Ende;
[ØWØ]  985:     Exit;
[ZNØ]  986:   End;
[DXØ]  987:   {$I-}
[9X1]  988:   ChDir (Pfad);
[BXØ]  989:   {$I+}
[MA6]  99Ø:   If IOResult <> Ø Then Fehlermeldung (4);
[LV1]  991:   CursorOff;
[ZTØ]  992:   Ende;
[ZNØ]  993: End;
[ØØØ]  994:
[W67]  995: {-- Hauptprogramm Subdirectory anlegen --}
[ØØØ]  996:
```

```
[343]  997: Procedure MakeDir;
[MUØ]  998: Begin
[G84]  999:   TextColors (Black,White);
[DB1] 1ØØØ:   ClrScr;
[F55] 1ØØ1:   xWindow (' S U B D I R E C T O R Y    '+
[D83] 1ØØ2:            'A N L E G E N ',1,1,8Ø,25);
[WE1] 1ØØ3:   AktPath;
[MN1] 1ØØ4:   CursorOn;
[G84] 1ØØ5:   TextColors (Black,White);
[486] 1ØØ6:   XWindow (' Bitte Namen eingeben, ggf. '+
[M44] 1ØØ7:            'mit Laufwerk und Pfad : ',
[HE1] 1ØØ8:            3,15,77,18);
[H72] 1ØØ9:   GotoXY ( 3, 2);
[XH2] 1Ø1Ø:   ReadLn (SubDir);
[MØ4] 1Ø11:   Test := Copy (SubDir,1,2);
[6X1] 1Ø12:   BLW (Test);
[H82] 1Ø13:   If Fehler Then
[MUØ] 1Ø14:   Begin
[ZTØ] 1Ø15:     Ende;
[ØWØ] 1Ø16:     Exit;
[ZNØ] 1Ø17:   End;
[DXØ] 1Ø18:   {$I-}
[ZC2] 1Ø19:     MkDir (SubDir);
[BXØ] 1Ø2Ø:   {$I+}
[HA6] 1Ø21:   If IOResult <> Ø Then Fehlermeldung (3);
[LV1] 1Ø22:   CursorOff;
[ZTØ] 1Ø23:   Ende;
[ZNØ] 1Ø24: End;
[ØØØ] 1Ø25:
[BC7] 1Ø26: {-- Hauptprogramm Subdirectory löschen --}
[ØØØ] 1Ø27:
[WH3] 1Ø28: Procedure ReMakeDir;
[MUØ] 1Ø29: Begin
[NR5] 1Ø3Ø:   For a := Ø To 255 Do Speicher[a] := '';
[DB1] 1Ø31:   ClrScr;
[F55] 1Ø32:   xWindow (' S U B D I R E C T O R Y    '+
[FC3] 1Ø33:            'L Ö S C H E N ',1,1,8Ø,25);
[WE1] 1Ø34:   AktPath;
[MN1] 1Ø35:   CursorOn;
[MR2] 1Ø36:   Window (2,2,79,23);
[W82] 1Ø37:   GotoXY (26, 1);
[X96] 1Ø38:   Writeln ('Verzeichnisse im Standard'+
[CV1] 1Ø39:            'laufwerk');
[2B3] 1Ø4Ø:   GetDir (Ø,AlterPfad);
[5P6] 1Ø41:  { aktuelles Verzeichnis speichern       }
[C75] 1Ø42:   Speicher[Ø] := AlterPfad[1]+':';
[P57] 1Ø43:  { Startverzeichnis in SpeicherØ sichern }
[W63] 1Ø44:   ChDir (Speicher[Ø]);
[ZD6] 1Ø45:  { ins Stammverzeichnis wechseln         }
[UT2] 1Ø46:   Durchsuchen := Ø;
[UL6] 1Ø47:  { zu durchsuchendes Verzeichnis         }
[752] 1Ø48:   Gefunden := Ø;
[PW5] 1Ø49:  { gefundene Verzeichnisse               }
[U53] 1Ø5Ø:   Verzeichnisse := 1;
[7E6] 1Ø51:  { Anzahl gelesener Verzeichnisse        }
[3P1] 1Ø52:   Files := Ø;
[9Ø6] 1Ø53:  { Anzahl gefundener Dateien             }
[M11] 1Ø54:   Suche;
[3M6] 1Ø55:  { Directory-Array initialisieren        }
[TG3] 1Ø56:   VerzeichnisAusgabe;
[LU2] 1Ø57:   ChDir (AlterPfad);
[VX4] 1Ø58:  { wieder ins ursprüngliche
[9F5] 1Ø59:    Verzeichnis zurück                    }
[486] 1Ø6Ø:   XWindow (' Bitte Namen eingeben, ggf. '+
[M44] 1Ø61:            'mit Laufwerk und Pfad : ',
[7E1] 1Ø62:            3,21,77,24);
[H72] 1Ø63:   GotoXY ( 3, 2);
[XH2] 1Ø64:   ReadLn (SubDir);
[MØ4] 1Ø65:   Test := Copy (SubDir,1,2);
[6X1] 1Ø66:   BLW (Test);
[H82] 1Ø67:   If Fehler Then
[MUØ] 1Ø68:   Begin
[ZTØ] 1Ø69:     Ende;
[ØWØ] 1Ø7Ø:     Exit;
[ZNØ] 1Ø71:   End;
[DXØ] 1Ø72:   {$I-}
[FD2] 1Ø73:     RmDir (SubDir);
[BXØ] 1Ø74:   {$I+}
[MA6] 1Ø75:   If IOResult <> Ø Then Fehlermeldung (4);
[LV1] 1Ø76:   CursorOff;
[ZTØ] 1Ø77:   Ende;
[ZNØ] 1Ø78: End;
[ØØØ] 1Ø79:
[665] 1Ø8Ø: {--- Hauptprogramm verstecktes
[PH5] 1Ø81:      Subdirectory anlegen ---------------}
[ØØØ] 1Ø82:
[U64] 1Ø83: Procedure MakeHiddenDir;
[RNØ] 1Ø84: Var
[ØT2] 1Ø85:   Reg : Registers;
[MUØ] 1Ø86: Begin
[DB1] 1Ø87:   ClrScr;
[YØ5] 1Ø88:   xWindow (' V E R S T E C K T E S   S '+
[6K3] 1Ø89:            'U B D I R E C T O R Y   A N'+
[Z33] 1Ø9Ø:            ' L E G E N ',1,1,8Ø,25);
[WE1] 1Ø91:   AktPath;
[KF1] 1Ø92:   Warnung;
[MN1] 1Ø93:   CursorOn;
[486] 1Ø94:   XWindow (' Bitte Namen eingeben, ggf. '+
[M44] 1Ø95:            'mit Laufwerk und Pfad : ',
[1E1] 1Ø96:            3,18,77,21);
[H72] 1Ø97:   GotoXY ( 3, 2);
[XH2] 1Ø98:   ReadLn (SubDir);
[MØ4] 1Ø99:   Test := Copy (SubDir,1,2);
[6X1] 11ØØ:   BLW (Test);
[H82] 11Ø1:   If Fehler Then
[MUØ] 11Ø2:   Begin
[ZTØ] 11Ø3:     Ende;
[ØWØ] 11Ø4:     Exit;
[ZNØ] 11Ø5:   End;
[DXØ] 11Ø6:   {$I-}
[ZC2] 11Ø7:     MkDir (SubDir);
[BXØ] 11Ø8:   {$I+}
[EA6] 11Ø9:   If IOResult <> Ø Then Fehlermeldung (2);
[973] 111Ø:   SubDir := SubDir + #Ø;
[6Z3] 1111:   Reg.DS := Seg (SubDir[1]);
[YZ3] 1112:   Reg.DX := Ofs (SubDir[1]);
[MX1] 1113:   Reg.CX := $Ø2;
[PN1] 1114:   Reg.AL := 1;
[GT1] 1115:   Reg.AH := 67;
[8T1] 1116:   MsDos (Reg);
[LV1] 1117:   CursorOff;
[ZTØ] 1118:   Ende;
[ZNØ] 1119: End;
[ØØØ] 112Ø:
[R47] 1121: {--- Hauptprogramm Attribute setzen -----}
[ØØØ] 1122:
[G73] 1123: Procedure SetAttr;
[MUØ] 1124: Begin
[DB1] 1125:   ClrScr;
[V75] 1126:   xWindow (' D A T E I A T T R I B U T E'+
[LL3] 1127:            '   S E T Z E N ',1,1,8Ø,25);
[WE1] 1128:   AktPath;
[Y83] 1129:   DateiEinlesen ('*.*');
[H82] 113Ø:   If Fehler Then
[MUØ] 1131:   Begin
[ZTØ] 1132:     Ende;
[ØWØ] 1133:     Exit;
[ZNØ] 1134:   End;
[4R1] 1135:   WaehleAus;
[H82] 1136:   If Fehler Then
[MUØ] 1137:   Begin
[ZTØ] 1138:     Ende;
[ØWØ] 1139:     Exit;
[ZNØ] 114Ø:   End;
[P31] 1141:   Repeat
[P56] 1142:     xWindow (' Bitte wählen ',22,14,56,24);
[NE5] 1143:     WriteXY ( 2, 2,'1 = Schreibschutz'+
[2N1] 1144:               '                      ');
[UE5] 1145:     WriteXY ( 2, 3,'2 = Schreibschutz'+
[9D2] 1146:               ' aufheben             ');
[W55] 1147:     WriteXY ( 2, 4,'3 = verstecken    '+
[2N1] 1148:               '                      ');
[Ø95] 1149:     WriteXY ( 2, 5,'4 = sichtbar mach'+
[9Y1] 115Ø:               'en                    ');
[VG5] 1151:     WriteXY ( 2, 6,'5 = als Systemdat'+
[653] 1152:               'ei kennzeichnen       ');
[5C5] 1153:     WriteXY ( 2, 7,'6 = Archivbit set'+
[KØ2] 1154:               'zen                   ');
[ØE5] 1155:     WriteXY ( 2, 8,'7 = Archivbit lös'+
[C12] 1156:               'chen                  ');
[ED5] 1157:     WriteXY ( 2, 9,'8 = Attribute nic'+
[LR2] 1158:               'ht ändern             ');
[E82] 1159:     GotoXY  (4Ø, 6);
[EP1] 116Ø:     ReadLn (a);
[T53] 1161:   Until IOResult = Ø;
[DXØ] 1162:   {$I-}
[S91] 1163:   Case a Of
[Ø64] 1164:    1 : SetFAttr (f,ReadOnly);
[BC4] 1165:    2 : SetFAttr (f,ReadOnly-1);
[7S3] 1166:    3 : SetFAttr (f,Hidden);
[LØ4] 1167:    4 : SetFAttr (f,Hidden-2);
[534] 1168:    5 : SetFAttr (f,SysFile);
[EØ4] 1169:    6 : SetFAttr (f,Archive);
[BF4] 117Ø:    7 : SetFAttr (f,Archive -32);
[691] 1171:    5 : Begin
[V42] 1172:          FullWindow;
[G84] 1173:          TextColors (Black,White);
[DB1] 1174:          ClrScr;
[ØWØ] 1175:          Exit;
[ZNØ] 1176:        End;
[ZNØ] 1177:   End;
[BXØ] 1178:   {$I+}
[UA6] 1179:   If IOResult <> Ø Then Fehlermeldung (6);
[ZTØ] 118Ø:   Ende;
[ZNØ] 1181: End;
[ØØØ] 1182:
[KN5] 1183: {--- Hauptprogramm Diskettendatei
```

Listing 3. Die Unit »disk.pas« stellt Datei- und Verzeichnis-Funktionen zur Verfügung

```
[ØU4] 1184:       umhängen ---------------------------}
[ØØØ] 1185:
[DD3] 1186: Procedure MoveFile;
[MUØ] 1187: Begin
[DB1] 1188:   ClrScr;
[K45] 1189:   xWindow (' V E R Z E I C H N I S E I '+
[HF3] 119Ø:           'N T R A G  E I N E R  D A'+
[6C3] 1191:           ' T E I  Ä N D E R N ',
[9C1] 1192:           1,1,8Ø,25);
[WE1] 1193:   AktPath;
[MN1] 1194:   CursorOn;
[2Z4] 1195:   XWindow (' H i n w e i s ',3,3,77,13);
[X26] 1196:   WriteXY (1Ø, 2,'Mit dieser Funktion '+
[YS4] 1197:           'haben Sie folgende Möglich'+
[AT1] 1198:           'keiten:');
[RS5] 1199:   WriteXY ( 5, 4,'1.) Nur umbenennen ='+
[HN4] 12ØØ:           ' Gleicher Pfad und untersc'+
[8T3] 12Ø1:           'hiedliche Dateinamen');
[BM5] 12Ø2:   WriteXY ( 5, 5,'2.) Nur umhängen   ='+
[3P4] 12Ø3:           ' Unterschiedliche Pfade un'+
[NL3] 12Ø4:           'd gleiche Dateinamen');
[LL5] 12Ø5:   WriteXY ( 5, 6,'3.) umbenennen und u'+
[PS4] 12Ø6:           'mhängen = Unterschiedliche'+
[PY3] 12Ø7:           ' Pfade und Dateinamen');
[XF5] 12Ø8:   WriteXY ( 8, 8,'A C H T U N G : Beid'+
[4Y4] 12Ø9:           'e Laufwerksangaben müssen '+
[H13] 121Ø:           'identisch sein!');
[296] 1211:   xWindow (' Bitte kompletten Pfad der'+
[KZ2] 1212:           ' Datei eingeben ! ',
[HE1] 1213:           3,15,77,18);
[H72] 1214:   GotoXY ( 3, 2);
[252] 1215:   ReadLn (Pfad);
[ZK3] 1216:   Test := Copy (Pfad,1,2);
[6X1] 1217:   BLW (Test);
[H82] 1218:   If Fehler Then
[MUØ] 1219:   Begin
[ZTØ] 122Ø:     Ende;
[ØWØ] 1221:     Exit;
[ZNØ] 1222:   End;
[LF2] 1223:   Assign (f,Pfad);
[2P6] 1224:   xWindow (' Bitte kompletten NEUEN Pfad'+
[BK3] 1225:           ' der Datei eingeben ! ',
[3E1] 1226:           3,2Ø,77,23);
[H72] 1227:   GotoXY ( 3, 2);
[AB2] 1228:   Readln (Pfad2);
[KR3] 1229:   Test := Copy (Pfad2,1,2);
[6X1] 123Ø:   BLW (Test);
[H82] 1231:   If Fehler Then
[MUØ] 1232:   Begin
[ZTØ] 1233:     Ende;
[ØWØ] 1234:     Exit;
[ZNØ] 1235:   End;
[DXØ] 1236:   {$I-}
[8K2] 1237:     Rename (f,Pfad2);
[BXØ] 1238:   {$I+}
[MA6] 1239:   If IOResult <> Ø Then Fehlermeldung (4);
[ZTØ] 124Ø:   Ende;
[ZNØ] 1241: End;
[ØØØ] 1242:
[577] 1243: {--- Hauptprogramm Dateizeit ändern -----}
[ØØØ] 1244:
[8X3] 1245: Procedure SetFileTime;
[RNØ] 1246: Var
[LY2] 1247:   Eingabe : Integer;
[MUØ] 1248: Begin
[DB1] 1249:   ClrScr;
[N65] 125Ø:   xWindow (' D A T E I D A T U M  U N D'+
[4W3] 1251:           '  - Z E I T  Ä N D E R N ',
[9C1] 1252:           1,1,8Ø,25);
[WE1] 1253:   AktPath;
[Y83] 1254:   DateiEinlesen ('*.*');
[H82] 1255:   If Fehler Then
[MUØ] 1256:   Begin
[ZTØ] 1257:     Ende;
[ØWØ] 1258:     Exit;
[ZNØ] 1259:   End;
[4R1] 126Ø:   WaehleAus;
[H82] 1261:   If Fehler Then
[MUØ] 1262:   Begin
[ZTØ] 1263:     Ende;
[ØWØ] 1264:     Exit;
[ZNØ] 1265:   End;
[MR2] 1266:   Window (2,2,79,23);
[DB1] 1267:   ClrScr;
[NF1] 1268:   Reset (f);
[RA6] 1269:   If IOResult <> Ø Then Fehlermeldung (5);
[922] 127Ø:   GetFTime (F,Zeit);
[MH3] 1271:   UnpackTime (Zeit,DT);
[2N5] 1272:   xWindow (' Aktuelle Dateidaten : ',
[GC1] 1273:           2Ø,3,6Ø,6);
[M72] 1274:   GotoXY ( 4, 2);
[9P4] 1275:   Write (Dateiname,'  ',DT.Day,'.',
[ØX3] 1276:          DT.Month,'.',DT.Year,'  ',
[2P4] 1277:          DT.Hour,':',DT.Min,':',DT.Sec);
[UA6] 1278:   xWindow (' Bitte neue Daten eingeben ',
[MC1] 1279:           2Ø,7,6Ø,2Ø);
[GW3] 128Ø:   WriteXY (1Ø, 4,'Tag    :  ');
[M24] 1281:   WriteXY (1Ø, 5,'Monat  :  ');
[SX3] 1282:   WriteXY (1Ø, 6,'Jahr   :  ');
[G74] 1283:   WriteXY (1Ø, 8,'Stunde :  ');
[H64] 1284:   WriteXY (1Ø, 9,'Minute :  ');
[HA4] 1285:   WriteXY (1Ø,1Ø,'Sekunde:  ');
[MN1] 1286:   CursorOn;
[P31] 1287:   Repeat
[Z82] 1288:     GotoXY (25, 4);
[DXØ] 1289:     {$I-}
[SB2] 129Ø:       Read (Eingabe);
[BXØ] 1291:     {$I+}
[9U3] 1292:   Until (IOResult = Ø) AND
[KE4] 1293:         (Eingabe > Ø) And (Eingabe < 32);
[GR2] 1294:   DT.Day := Eingabe;
[P31] 1295:   Repeat
[192] 1296:     GotoXY (25, 5);
[DXØ] 1297:     {$I-}
[SB2] 1298:       Read (Eingabe);
[BXØ] 1299:     {$I+}
[9U3] 13ØØ:   Until (IOResult = Ø) AND
[GE4] 1301:         (Eingabe > Ø) And (Eingabe < 13);
[693] 1302:   DT.Month := Eingabe;
[P31] 1303:   Repeat
[392] 1304:     GotoXY (25, 6);
[DXØ] 1305:     {$I-}
[SB2] 1306:       Read (Eingabe);
[BXØ] 1307:     {$I+}
[9U3] 1308:   Until (IOResult = Ø) AND
[WU2] 1309:         (Eingabe > 198Ø) And
[692] 131Ø:         (Eingabe < 21ØØ);
[K23] 1311:   DT.Year := Eingabe;
[P31] 1312:   Repeat
[792] 1313:     GotoXY (25, 8);
[DXØ] 1314:     {$I-}
[SB2] 1315:       Read (Eingabe);
[BXØ] 1316:     {$I+}
[9U3] 1317:   Until (IOResult = Ø) AND
[ØT4] 1318:         (Eingabe >= Ø) And (Eingabe < 24);
[H43] 1319:   DT.Hour := Eingabe;
[P31] 132Ø:   Repeat
[992] 1321:     GotoXY (25, 9);
[DXØ] 1322:     {$I-}
[SB2] 1323:       Read (Eingabe);
[BXØ] 1324:     {$I+}
[9U3] 1325:   Until (IOResult = Ø) AND
[CT4] 1326:         (Eingabe >= Ø) And (Eingabe < 6Ø);
[8R2] 1327:   DT.Min := Eingabe;
[P31] 1328:   Repeat
[892] 1329:     GotoXY (25,1Ø);
[DXØ] 133Ø:     {$I-}
[SB2] 1331:       Read (Eingabe);
[BXØ] 1332:     {$I+}
[9U3] 1333:   Until (IOResult = Ø) AND
[CT4] 1334:         (Eingabe >= Ø) And (Eingabe < 6Ø);
[5P2] 1335:   DT.Sec := Eingabe;
[453] 1336:   PackTime (DT,Zeit);
[1Ø3] 1337:   SetFTime (f,Zeit);
[LV1] 1338:   CursorOff;
[CE1] 1339:   Close (f);
[ZTØ] 134Ø:   Ende;
[ZNØ] 1341: End;
[ØØØ] 1342:
[KN5] 1343: {--- Hauptprogramm Diskettendatei
[PV4] 1344:       umbenennen -------------------------}
[ØØØ] 1345:
[KN3] 1346: Procedure RenameFile;
[MUØ] 1347: Begin
[DB1] 1348:   ClrScr;
[T25] 1349:   xWindow (' D A T E I  U M B E N E N N'+
[F92] 135Ø:           ' E N ',1,1,8Ø,25);
[WE1] 1351:   AktPath;
[MN1] 1352:   CursorOn;
[GA6] 1353:   XWindow (' Bitte Dateinamen, ggf. mit '+
[RZ4] 1354:            'Laufwerks- und Pfadangabe ei'+
[Z33] 1355:            'ngeben ! ',3,15,77,18);
[H72] 1356:   GotoXY ( 3, 2);
[252] 1357:   ReadLn (Pfad);
[ZK3] 1358:   Test := Copy (Pfad,1,2);
[6X1] 1359:   BLW (Test);
[H82] 136Ø:   If Fehler Then
[MUØ] 1361:   Begin
[ZTØ] 1362:     Ende;
[ØWØ] 1363:     Exit;
[ZNØ] 1364:   End;
[LF2] 1365:   Assign (f,Pfad);
[XU5] 1366:   xWindow (' Bitte NEUEN Dateinamen, '+
[4G4] 1367:            'ggf. mit Laufwerks- und P'+
[VH3] 1368:            'fadangabe eingeben ! ',
[3E1] 1369:            3,2Ø,77,23);
[H72] 137Ø:   GotoXY ( 3, 2);
[AB2] 1371:   Readln (Pfad2);
[KR3] 1372:   Test := Copy (Pfad2,1,2);
[6X1] 1373:   BLW (Test);
[H82] 1374:   If Fehler Then
[MUØ] 1375:   Begin
```

```
[ZTØ] 1376:      Ende;
[ØWØ] 1377:      Exit;
[ZNØ] 1378:    End;
[DXØ] 1379:    {$I-}
[8K2] 138Ø:      Rename (f,Pfad2);
[BXØ] 1381:    {$I+}
[MA6] 1382:    If IOResult <> Ø Then Fehlermeldung (4);
[ZTØ] 1383:    Ende;
[ZNØ] 1384: End;
[ØØØ] 1385:
[KN5] 1386: {--- Hauptprogramm Diskettendatei
[HZ4] 1387:      löschen ---------------------------}
[ØØØ] 1388:
[YM3] 1389: Procedure DeleteFile;
[MUØ] 139Ø: Begin
[DB1] 1391:    ClrScr;
[D55] 1392:    xWindow (' D A T E I   L Ö S C H E N ',
[9C1] 1393:             1,1,8Ø,25);
[WE1] 1394:    AktPath;
[MN1] 1395:    CursorOn;
[GA6] 1396:    XWindow (' Bitte Dateinamen, ggf. mit '+
[RZ4] 1397:             'Laufwerks- und Pfadangabe ei'+
[Z33] 1398:             'ngeben ! ',3,15,77,18);
[H72] 1399:    GotoXY ( 3, 2);
[252] 14ØØ:    ReadLn (Pfad);
[ZK3] 14Ø1:    Test := Copy (Pfad,1,2);
[6X1] 14Ø2:    BLW (Test);
[H82] 14Ø3:    If Fehler Then
[MUØ] 14Ø4:    Begin
[ZTØ] 14Ø5:      Ende;
[ØWØ] 14Ø6:      Exit;
[ZNØ] 14Ø7:    End;
[LF2] 14Ø8:    Assign (f,Pfad);
[DXØ] 14Ø9:    {$I-}
[AD1] 141Ø:      Erase (f);
[BXØ] 1411:    {$I+}
[MA6] 1412:    If IOResult <> Ø Then Fehlermeldung (4);
[ZTØ] 1413:    Ende;
[ZNØ] 1414: End;
[ØØØ] 1415:
[4T6] 1416: {--- Hauptprogramm Datei kopieren -------}
[ØØØ] 1417:
[ND3] 1418: Procedure CopyFile;
[RNØ] 1419: Var
[4HØ] 142Ø:    QF,
[ZY1] 1421:    ZF       : File;
[3L1] 1422:    BytesEin,
[SP2] 1423:    BytesAus : Word;
[345] 1424:    Puffer   : Array[1..2Ø48] of Byte;
[MUØ] 1425: Begin
[DB1] 1426:    ClrScr;
[W55] 1427:    xWindow (' D A T E I   K O P I E R E N ',
[9C1] 1428:             1,1,8Ø,25);
[WE1] 1429:    AktPath;
[MN1] 143Ø:    CursorOn;
[D43] 1431:    xWindow ('',3,5,77,8);
[BH5] 1432:    WriteXY ( 3, 2,'Bitte Quelldatei '+
[K72] 1433:             'eingeben : ');
[252] 1434:    ReadLn (Pfad);
[ZK3] 1435:    Test := Copy (Pfad,1,2);
[6X1] 1436:    BLW (Test);
[H82] 1437:    If Fehler Then
[MUØ] 1438:    Begin
[ZTØ] 1439:      Ende;
[ØWØ] 144Ø:      Exit;
[ZNØ] 1441:    End;
[344] 1442:    If FindFile(Pfad) <> Ø Then
[MUØ] 1443:    Begin
[8W2] 1444:      Fehlermeldung (1);
[ZTØ] 1445:      Ende;
[ØWØ] 1446:      Exit;
[ZNØ] 1447:    End;
[PB3] 1448:    xWindow ('',3,15,77,18);
[EB5] 1449:    WriteXY ( 3, 2,'Bitte Zieldatei '+
[K72] 145Ø:             'eingeben : ');
[AB2] 1451:    ReadLn (Pfad2);
[KR3] 1452:    Test := Copy (Pfad2,1,2);
[6X1] 1453:    BLW (Test);
[H82] 1454:    If Fehler Then
[MUØ] 1455:    Begin
[ZTØ] 1456:      Ende;
[ØWØ] 1457:      Exit;
[ZNØ] 1458:    End;
[B13] 1459:    If Pfad = Pfad2 Then
[MUØ] 146Ø:    Begin
[K23] 1461:      Fehlermeldung (13);
[ZTØ] 1462:      Ende;
[ØWØ] 1463:      Exit;
[ZNØ] 1464:    End;
[Z14] 1465:    If FindFile(Pfad2) = Ø Then
[MUØ] 1466:    Begin
[G23] 1467:      Fehlermeldung (12);
[WD3] 1468:      xWindow ('',2Ø,1Ø,6Ø,13);
[116] 1469:      WriteXY ( 3, 2,'ÜBERSCHREIBEN (J/N) ');
[P31] 147Ø:      Repeat
[DXØ] 1471:        {$I-}
[Y81] 1472:          Read (c);
[BXØ] 1473:        {$I+}
[GS3] 1474:      Until (Upcase(c) = 'J') Or
[YK2] 1475:            (Upcase(c) = 'N');
[LV1] 1476:      CursorOff;
[2B3] 1477:      If Upcase (c) = 'N' Then
[MUØ] 1478:      Begin
[ZTØ] 1479:        Ende;
[ØWØ] 148Ø:        Exit;
[ZNØ] 1481:      End;
[ZNØ] 1482:    End;
[MR2] 1483:    Window (2,2,79,23);
[DB1] 1484:    ClrScr;
[2N2] 1485:    Assign (QF,Pfad);
[CØ2] 1486:    Reset  (QF,1);
[AU2] 1487:    Assign (ZF,Pfad2);
[EK2] 1488:    Rewrite (ZF,1);
[X72] 1489:    GotoXY ( 5, 5);
[1T3] 149Ø:    Write ('Kopiere .......');
[182] 1491:    GotoXY ( 5, 7);
[373] 1492:    Write ('von   ',Pfad);
[582] 1493:    GotoXY ( 5, 9);
[DB3] 1494:    Write ('nach  ',Pfad2);
[P31] 1495:    Repeat
[XY5] 1496:      BlockRead (QF,Puffer,SizeOf(Puffer),
[ES1] 1497:                 BytesEin);
[EA5] 1498:      BlockWrite (ZF,Puffer,BytesEin,
[RT1] 1499:                  BytesAus);
[HS3] 15ØØ:    Until (BytesEin = Ø) Or
[ZV3] 15Ø1:          (BytesEin <> BytesAus);
[GP1] 15Ø2:    Close (ZF);
[7P1] 15Ø3:    Close (QF);
[VR1] 15Ø4:    Delay (2ØØØ);
[ZTØ] 15Ø5:  Ende;
[ZNØ] 15Ø6: End;
[ØØØ] 15Ø7:
[4R6] 15Ø8: {--- Hauptprogramm Datei anzeigen -------}
[ØØØ] 15Ø9:
[6F3] 151Ø: Procedure ShowFile;
[RNØ] 1511: Var
[NL2] 1512:    Zeile : String;
[VZØ] 1513:    Seite,
[V22] 1514:    Laenge: Real;
[MUØ] 1515: Begin
[DB1] 1516:    ClrScr;
[G45] 1517:    xWindow (' D A T E I   A N Z E I G E N ',
[9C1] 1518:             1,1,8Ø,25);
[WE1] 1519:    AktPath;
[MN1] 152Ø:    CursorOn;
[D43] 1521:    xWindow ('',3,5,77,8);
[UE5] 1522:    WriteXY ( 3, 2,'Bitte Dateinamen '+
[K72] 1523:             'eingeben : ');
[252] 1524:    ReadLn (Pfad);
[ZK3] 1525:    Test := Copy (Pfad,1,2);
[6X1] 1526:    BLW (Test);
[H82] 1527:    If Fehler Then
[MUØ] 1528:    Begin
[ZTØ] 1529:      Ende;
[ØWØ] 153Ø:      Exit;
[ZNØ] 1531:    End;
[LV1] 1532:    CursorOff;
[344] 1533:    If FindFile(Pfad) <> Ø Then
[MUØ] 1534:    Begin
[8W2] 1535:      Fehlermeldung (1);
[ZTØ] 1536:      Ende;
[ØWØ] 1537:      Exit;
[ZNØ] 1538:    End;
[FG2] 1539:    Assign (t,Pfad);
[3G1] 154Ø:    Reset (t);
[V42] 1541:    FullWindow;
[382] 1542:    GotoXY ( 1,25);
[S32] 1543:    Write (' ':78);
[CF5] 1544:    WriteXY (2Ø,25,'BLÄTTERN DURCH '+
[5C4] 1545:             'BELIEBIGEN TASTENDRUCK');
[MR2] 1546:    Window (2,2,79,23);
[DB1] 1547:    ClrScr;
[SN1] 1548:    Seite := Ø;
[R43] 1549:    While Not EOF(t) Do
[MUØ] 155Ø:    Begin
[7R2] 1551:      ReadLn (t,Zeile);
[7G5] 1552:      Laenge := Int(Length(Zeile) / 78) + 1;
[1R3] 1553:      Seite := Seite + Laenge;
[3R2] 1554:      WriteLn (Zeile);
[UN2] 1555:      If Seite > 19 Then
[MUØ] 1556:      Begin
[SN1] 1557:        Seite := Ø;
[P31] 1558:        Repeat
```

Listing 3. Die Unit »disk.pas« stellt Datei- und Verzeichnis-Funktionen zur Verfügung

```
[2Z1] 1559:          c := ReadKey;
[R12] 156Ø:        Until c <> #Ø;
[ZNØ] 1561:      End;
[ZNØ] 1562:    End;
[AF1] 1563:    Close(t);
[P31] 1564:    Repeat
[2Z1] 1565:      c := ReadKey;
[R12] 1566:    Until c <> #Ø;
[ZTØ] 1567:    Ende;
[ZNØ] 1568: End;
[ØØØ] 1569:
[6P6] 157Ø: {--- Hauptprogramm Datei drucken --------}
[ØØØ] 1571:
[4L3] 1572: Procedure PrintFile;
[RNØ] 1573: Var
[7L2] 1574:    ZNr   : Boolean;
[NL2] 1575:    Zeile : String;
[T12] 1576:    Anz   : Word;
[EW2] 1577:    Reg   : Registers;
[MUØ] 1578: Begin
[DB1] 1579:    ClrScr;
[2G4] 158Ø:    xWindow (' D A T E I   A U S D R '+
[1V2] 1581:             'U C K E N ',1,1,8Ø,25);
[WE1] 1582:    AktPath;
[A42] 1583:    Reg.AX := $Ø2ØØ;
[WP1] 1584:    Reg.DX := Ø;
[X32] 1585:    Intr (23,Reg);
[174] 1586:    If (Reg.AX And $8ØØ) <> Ø Then
[MUØ] 1587:    Begin
[W23] 1588:      Fehlermeldung (18);
[ZTØ] 1589:      Ende;
[ØWØ] 159Ø:      Exit;
[ZNØ] 1591:    End;
[MN1] 1592:    CursorOn;
[2B3] 1593:    xWindow ('',3,1Ø,77,13);
[UE5] 1594:    WriteXY ( 3, 2,'Bitte Dateinamen '+
[K72] 1595:              'eingeben : ');
[252] 1596:    ReadLn (Pfad);
[ZK3] 1597:    Test := Copy (Pfad,1,2);
[6X1] 1598:    BLW (Test);
[H82] 1599:    If Fehler Then
[MUØ] 16ØØ:    Begin
[ZTØ] 16Ø1:      Ende;
[ØWØ] 16Ø2:      Exit;
[ZNØ] 16Ø3:    End;
[LV1] 16Ø4:    CursorOff;
[344] 16Ø5:    If FindFile(Pfad) <> Ø Then
[MUØ] 16Ø6:    Begin
[FT2] 16Ø7:      Fehlermeldung(1);
[ZTØ] 16Ø8:      Ende;
[ØWØ] 16Ø9:      Exit;
[ZNØ] 161Ø:    End;
[PB3] 1611:    xWindow ('',3,15,77,18);
[5A6] 1612:    WriteXY ( 3, 2,'Sollen Zeilennummern '+
[VA4] 1613:             'gedruckt werden < J/N > ? ');
[P31] 1614:    Repeat
[K82] 1615:      GotoXY (5Ø, 2);
[KP1] 1616:      ReadLn (c);
[LY3] 1617:    Until (Upcase (c) = 'J') or
[RP2] 1618:          (Upcase (c) = 'N');
[7B5] 1619:    If Upcase (c) = 'J' Then ZNr := True
[6Z2] 162Ø:    Else Znr := False;
[FG2] 1621:    Assign (t,Pfad);
[3G1] 1622:    Reset (t);
[MR2] 1623:    Window (2,2,79,23);
[DB1] 1624:    ClrScr;
[N54] 1625:    WriteXY ( 27,1Ø,'B I T T E   '+
[TY1] 1626:             'W A R T E N !');
[3W5] 1627:    WriteXY ( 27,15,'   Ausdruck läuft');
[6A1] 1628:    Anz := 1;
[R43] 1629:    While Not EOF(t) Do
[MUØ] 163Ø:    Begin
[7R2] 1631:      ReadLn (t,Zeile);
[TK5] 1632:      If ZNr Then Write (Lst,'<',Anz:4,'>');
[BY3] 1633:      WriteLn (Lst,'  ',Zeile);
[GK1] 1634:      Inc (Anz);
[ZNØ] 1635:    End;
[AF1] 1636:    Close(t);
[ZTØ] 1637:    Ende;
[ZNØ] 1638: End;
[ØØØ] 1639:
[MT6] 164Ø: {--- Hauptprogramm Verzeichnisstruktur
[XØ5] 1641:      anzeigen (Tree) --------------------}
[ØØØ] 1642:
[TG3] 1643: Procedure ShowTree;
[MUØ] 1644: Begin
[RR2] 1645:    Dateien := False;
[NR5] 1646:    For a := Ø To 255 Do Speicher[a] := '';
[DB1] 1647:    ClrScr;
[765] 1648:    xWindow (' V E R Z E I C H N I S S T '+
[5G3] 1649:             'R U K T U R   A N Z E I G E'+
[ZØ2] 165Ø:             ' N ',1,1,8Ø,25);
[WE1] 1651:    AktPath;
[MN1] 1652:    CursorOn;
[P31] 1653:    Repeat
[D43] 1654:      xWindow ('',3,5,77,8);
[Ø44] 1655:      { Laufwerk einlesen   }
[445] 1656:      WriteXY ( 3, 2,'Bitte Laufwerk '+
[K72] 1657:                'eingeben : ');
[1V1] 1658:      ReadLn (LW);
[Ø24] 1659:    Until (Upcase (LW) >= 'A') And
[RX2] 166Ø:          (Upcase (LW) < 'G');
[VC2] 1661:    Test := LW + ':';
[6X1] 1662:    BLW (Test);
[H82] 1663:    If Fehler Then
[MUØ] 1664:    Begin
[ZTØ] 1665:      Ende;
[ØWØ] 1666:      Exit;
[ZNØ] 1667:    End;
[P31] 1668:    Repeat
[2B3] 1669:      xWindow ('',3,1Ø,77,13);
[A14] 167Ø:     { Mit Dateiausgabe??  }
[AC5] 1671:      WriteXY ( 3, 2,'Mit Ausgabe der '+
[SD3] 1672:               'Dateinamen (J/N) : ');
[SR1] 1673:      ReadLn (v);
[4V3] 1674:    Until (Upcase(v) = 'N') Or
[ML2] 1675:          (Upcase(v) = 'J');
[326] 1676:    If Upcase (v) = 'J' Then Dateien := True;
[LV1] 1677:    CursorOff;
[2T2] 1678:    Window (1,25,79,25);
[DB1] 1679:    ClrScr;
[XX5] 168Ø:    WriteXY (3Ø, 1,'BLÄTTERN MIT  <CR>');
[MR2] 1681:    Window (2,2,79,23);
[DB1] 1682:    ClrScr;
[2B3] 1683:    GetDir (Ø,AlterPfad);
[5P6] 1684:   { aktuelles Verzeichnis speichern        }
[CC3] 1685:    Speicher[Ø] := LW+':';
[P57] 1686:   { Startverzeichnis in SpeicherØ sichern }
[W63] 1687:    ChDir (Speicher[Ø]);
[ZD6] 1688:   { ins Stammverzeichnis wechseln          }
[UT2] 1689:    Durchsuchen := Ø;
[UL6] 169Ø:   { zu durchsuchendes Verzeichnis         }
[752] 1691:    Gefunden := Ø;
[PW5] 1692:   { gefundene Verzeichnisse               }
[U53] 1693:    Verzeichnisse := 1;
[7E6] 1694:   { Anzahl gelesener Verzeichnisse        }
[3P1] 1695:    Files := Ø;
[9Ø6] 1696:   { Anzahl gefundener Dateien             }
[M11] 1697:    Suche;
[3M6] 1698:   { Directory-Array initialisieren       }
[PL3] 1699:    If Dateien = False Then
[D93] 17ØØ:      VerzeichnisAusgabe
[NN6] 17Ø1:   { Ausgabe aller Verzeichnisse oder .... }
[XØ3] 17Ø2:      Else DateiAusgabe;
[697] 17Ø3:   { Ausgabe von Verzeichnissen UND Dateien }
[Y44] 17Ø4:    Repeat Until Keypressed;
[LU2] 17Ø5:    ChDir (AlterPfad);
[VX4] 17Ø6:   { wieder ins ursprüngliche
[9F5] 17Ø7:     Verzeichnis zurück                    }
[ZTØ] 17Ø8:    Ende;
[ZNØ] 17Ø9: End;
[ØØØ] 171Ø:
[527] 1711: {-- Hauptprogramm Dateieintrag suchen ---}
[ØØØ] 1712:
[AN3] 1713: Procedure SearchFile;
[MUØ] 1714: Begin
[RR2] 1715:    Dateien := False;
[NR5] 1716:    For a := Ø To 255 Do Speicher[a] := '';
[DB1] 1717:    ClrScr;
[T15] 1718:    xWindow (' D A T E I E I N T R A G     '+
[RØ3] 1719:             'S U C H E N ',1,1,8Ø,25);
[WE1] 172Ø:    AktPath;
[MN1] 1721:    CursorOn;
[P31] 1722:    Repeat
[D43] 1723:      xWindow ('',3,5,77,8);
[Ø44] 1724:  { Laufwerk einlesen   }
[445] 1725:      WriteXY ( 3, 2,'Bitte Laufwerk '+
[K72] 1726:                'eingeben : ');
[1V1] 1727:      ReadLn (LW);
[Ø24] 1728:    Until (Upcase (LW) >= 'A') And
[RX2] 1729:          (Upcase (LW) < 'G');
[VC2] 173Ø:    Test := LW + ':';
[6X1] 1731:    BLW (Test);
[H82] 1732:    If Fehler Then
[MUØ] 1733:    Begin
[ZTØ] 1734:      Ende;
[ØWØ] 1735:      Exit;
[ZNØ] 1736:    End;
[P31] 1737:    Repeat
[2B3] 1738:      xWindow ('',3,1Ø,77,13);
[CB5] 1739:      WriteXY ( 3, 2,'Bitte gesuchten '+
[214] 174Ø:               'Dateinamen eingeben : ');
[343] 1741:      ReadLn (Dateiname);
[ME3] 1742:    Until Dateiname <> '';
[DG4] 1743:    Test := Copy (Dateiname,1,2);
[6X1] 1744:    BLW (Test);
[H82] 1745:    If Fehler Then
[MUØ] 1746:    Begin
[ZTØ] 1747:      Ende;
[ØWØ] 1748:      Exit;
[ZNØ] 1749:    End;
```

```
[BP2] 175Ø:    For a := 1 To 12 Do
[A96] 1751:      Dateiname[a] := Upcase(Dateiname[a]);
[LV1] 1752:    CursorOff;
[2T2] 1753:    Window (1,25,79,25);
[DB1] 1754:    ClrScr;
[XC5] 1755:    WriteXY (3Ø, 1,'WEITER MIT  <CR>');
[MR2] 1756:    Window (2,2,79,23);
[DB1] 1757:    ClrScr;
[2B3] 1758:    GetDir (Ø,AlterPfad);
[5P6] 1759:   { aktuelles Verzeichnis speichern       }
[CC3] 176Ø:    Speicher[Ø] := LW+':';
[DC6] 1761:   { Startverzeichnis in Speicher1        }
[W63] 1762:    ChDir (Speicher[Ø]);
[ZD6] 1763:   { ins Stammverzeichnis wechseln        }
[UT2] 1764:    Durchsuchen := Ø;
[UL6] 1765:   { zu durchsuchendes Verzeichnis       }
[752] 1766:    Gefunden := Ø;
[PW5] 1767:   { gefundene Verzeichnisse             }
[U53] 1768:    Verzeichnisse := 1;
[7E6] 1769:   { Anzahl gelesener Verzeichnisse      }
[3P1] 177Ø:    Files := Ø;
[9Ø6] 1771:   { Anzahl gefundener Dateien           }
[M11] 1772:    Suche;
[3M6] 1773:   { Directory-Array initialisieren      }
[GS4] 1774:    GefundenAnzeigen(Dateiname);
[LU2] 1775:    ChDir (AlterPfad);
[VX4] 1776:   { wieder ins ursprüngliche
[9X3] 1777:     Verzeichnis zurück}
[Y44] 1778:    Repeat until Keypressed;
[ZTØ] 1779:    Ende;
[ZNØ] 178Ø: End;
[ØØØ] 1781:
[656] 1782:.{--- Hauptprogramm DOS-Kommandozeile
[4P4] 1783:       eingeben --------------------------}
[ØØØ] 1784:
[1N3] 1785: Procedure DosCommand;
[RNØ] 1786: Var
[754] 1787:    Parameter : String[64];
[MUØ] 1788: Begin
[VR2] 1789:    Window (1,22,8Ø,24);
[DB1] 179Ø:    ClrScr;
[MN1] 1791:    CursorOn;
[736] 1792:    xWindow (' Bitte DOS-Kommandozeile '+
[5E3] 1793:             'eingeben : ',1,22,8Ø,25);
[E72] 1794:    GotoXY ( 2, 2);
[U63] 1795:    ReadLn (Parameter);
[V42] 1796:    FullWindow;
[DB1] 1797:    ClrScr;
[ZN5] 1798:    Exec ('COMMAND.COM','/C '+Parameter);
[LV1] 1799:    CursorOff;
[XF1] 18ØØ:    Writeln;
[UG5] 18Ø1:    WriteXY (3Ø,25,'< BITTE BELIEBIGE '+
[C43] 18Ø2:              'TASTE DRÜCKEN >');
[Y44] 18Ø3:    Repeat Until KeyPressed;
[ZTØ] 18Ø4:    Ende;
[ZNØ] 18Ø5: End;
[ØØØ] 18Ø6:
[KNØ] 18Ø7: End.
```

Listing 3. Die Unit »disk.pas« stellt Datei- und Verzeichnis-Funktionen zur Verfügung

```
[Y65]    1: { Programmname: EXTERN.PAS    }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {           DOS International }
[ØØØ]    4:
[6Z2]    5: {  U N I T   E X T E R N
[X24]    6:    Autor: Karlheinz Büker
[ØØØ]    7:
[9E6]    8:    Enthält menügesteuerten Aufruf für
[D17]    9:    verschiedene MS-DOS-Utilities innerhalb
[D76]   1Ø:    von EASYDOS. Die benötigten MS-DOS-
[7C6]   11:    Programmdateien bitte im EASYDOS-Ver-
[1Ø7]   12:    zeichnis ablegen. Alternativ ist über die
[4A6]   13:    Menüfunktion "Extern-Pfad setzen" das
[DW6]   14:    Setzen eines expliziten Pfades möglich.
[EH5]   15:    EASYDOS sucht die entsprechenden
[9H5]   16:    Programme dann im angegebenen Ver-
[DV6]   17:    zeichnis. Im Quelltext sind jeweils die
[DX5]   18:    Programmnamen als .COM - Dateien vor-
[8D5]   19:    gegeben. Falls Ihre DOS-Version
[AS5]   2Ø:    die EXE-Versionen der benötigten
[K47]   21:    Utilities enthält, müssen die jeweiligen
[XH6]   22:    Zeilen entsprechend geändert werden !
[R85]   23:    Je nach DOS-Version können die
[8R6]   24:    Kommandozeilenparameter der Utilities
[LW5]   25:    differieren. In diesen Fällen muss
[BA6]   26:    dieses Unit entsprechend korrigiert
[TP5]   27:    werden! (Siehe DOS-Handbuch).          }
[ØØØ]   28:
[262]   29: Unit Extern;
[ØØØ]   3Ø:
[SK1]   31: INTERFACE
[ØØØ]   32:
[ASØ]   33: Uses
[AR3]   34:   Dos, Crt, Global, Disk;
[ØØØ]   35:
[RNØ]   36: Var
[864]   37:   ExternPfad : String[64];
[ØØØ]   38:
[NP3]   39: Procedure RunCommand;
[9M3]   4Ø: { zum DOS-Modus          }
[NN3]   41: Procedure FormatDisk;
[V64]   42: { Diskette formatieren   }
[DE3]   43: Procedure Diskcopy;
[WY3]   44: { Diskette kopieren      }
[6S3]   45: Procedure RunProgram;
[MU3]   46: { Programmaufruf         }
[4G3]   47: Procedure CheckDisk;
[AW3]   48: { Diskette prüfen        }
[SC4]   49: Procedure SetExternPath;
[9C4]   5Ø: { Pfad für Extern-Progr.}
[ØØØ]   51:
[4L2]   52: IMPLEMENTATION
[ØØØ]   53:
[W17]   54: {--- Pfad für externe Programme setzen --}
[ØØØ]   55:
[SC4]   56: Procedure SetExternPath;
[RNØ]   57: Var
[TV1]   58:   Test : Str2;
[MUØ]   59: Begin
[PK2]   6Ø:   ExternPfad := '';
[DB1]   61:   ClrScr;
[R35]   62:   XWindow (' P F A D   F Ü R   E X T E '+
[YH3]   63:            'R N E   D O S - P R O G R A'+
[253]   64:            ' M M E   S E T Z E N ',
[9C1]   65:            1,1,8Ø,25);
[WE1]   66:   AktPath;
[2V6]   67:   XWindow ('Bitte Pfad für externe DOS-'+
[KS4]   68:            'Programme eingeben, mit "\"'+
[D94]   69:            ' abschliessen : ',3,1Ø,77,13);
[MN1]   7Ø:   CursorOn;
[H72]   71:   GotoXY ( 3, 2);
[DA3]   72:   Readln (ExternPfad);
[LV1]   73:   CursorOff;
[ØR4]   74:   Test := Copy (ExternPfad,1,2);
[6X1]   75:   BLW (Test);
[H82]   76:   If Fehler Then
[MUØ]   77:   Begin
[PK2]   78:     ExternPfad := '';
[ZTØ]   79:     Ende;
[ØWØ]   8Ø:     Exit;
[ZNØ]   81:   End;
[7P5]   82:   Assign (f,ExternPfad+'COMMAND.COM');
[DXØ]   83:   {$I-}
[NF1]   84:     Reset (f);
[BXØ]   85:   {$I+}
[5A3]   86:   If IOResult <> Ø Then
[MUØ]   87:   Begin
[M23]   88:     Fehlermeldung (14);
[PK2]   89:     ExternPfad := '';
[ZTØ]   9Ø:     Ende;
[ØWØ]   91:     Exit;
[ZNØ]   92:   End;
[CE1]   93:   Close (f);
[ZTØ]   94:   Ende;
[ZNØ]   95: End;
[ØØØ]   96:
[Z77]   97: {-- vorübergehend in DOS-Modus wechseln -}
[ØØØ]   98:
[NP3]   99: Procedure RunCommand;
[MUØ]  1ØØ: Begin
[MN1]  1Ø1:   CursorOn;
[V42]  1Ø2:   FullWindow;
[DB1]  1Ø3:   ClrScr;
[7P5]  1Ø4:   Assign (f,ExternPfad+'COMMAND.COM');
[DXØ]  1Ø5:   {$I-}
[NF1]  1Ø6:     Reset (f);
[BXØ]  1Ø7:   {$I+}
[5A3]  1Ø8:   If IOResult <> Ø Then
[MUØ]  1Ø9:   Begin
[FU2]  11Ø:     Fehlermeldung(9);
[ØWØ]  111:     Exit;
[ZNØ]  112:   End;
[CE1]  113:   Close (f);
[NE5]  114:   Exec (ExternPfad+'COMMAND.COM','');
[5P3]  115:   If DosExitCode <> Ø Then
[C23]  116:     Fehlermeldung (1Ø);
[LV1]  117:   CursorOff;
[DB1]  118:   ClrScr;
[ZTØ]  119:   Ende;
[ZNØ]  12Ø: End;
```

Listing 4. Die Unit »extern.pas« wickelt menügesteuerte Aufrufe externer MS-DOS-Funktionen ab

```
[ØØØ]  121:
[F57]  122: {-- Hauptprogramm Diskette formatieren --}
[ØØØ]  123:
[NN3]  124: Procedure FormatDisk;
[RNØ]  125: Var
[4Ø2]  126:   a          : Byte;
[Y44]  127:   Parameter : String[12];
[SW1]  128:   c          : Char;
[5ZØ]  129:   Beids,
[9TØ]  13Ø:   Sys,
[FRØ]  131:   Kop,
[7E1]  132:   Kennung,
[DKØ]  133:   LW,
[ØU3]  134:   Sektoren  : String[2];
[MUØ]  135: Begin
[DB1]  136:   ClrScr;
[275]  137:   xWindow (' D I S K E T T E   F O R M A'+
[9E3]  138:           ' T I E R E N ',1,1,8Ø,25);
[BR5]  139:   Assign (f,ExternPfad+'FORMAT.COM');
[KA4]  14Ø:  { COM ggf. in EXE ändern ! }
[DXØ]  141:   {$I-}
[NF1]  142:     Reset (f);
[BXØ]  143:   {$I+}
[5A3]  144:   If IOResult <> Ø Then
[MUØ]  145:   Begin
[8Y2]  146:     Fehlermeldung(15);
[ØWØ]  147:     Exit;
[ZNØ]  148:   End;
[CE1]  149:   Close (f);
[9Ø4]  15Ø:   Parameter := '                    ';
[ZC2]  151:   Parameter := '';
[A31]  152:   LW := '';
[MN1]  153:   CursorOn;
[Z93]  154:   xWindow ('',1Ø,3,7Ø,23);
[526]  155:   WriteXY ( 3, 3,'In welchem Laufwerk f'+
[VM2]  156:            'ormatieren ? ');
[R76]  157:   WriteXY ( 3, 5,'Beide Seiten formatie'+
[N22]  158:            'ren  (J/N) ? ');
[SV5]  159:   WriteXY ( 3, 7,'8 oder 9 Sektoren / S'+
[XV1]  16Ø:            'pur        ? ');
[8E6]  161:   WriteXY ( 3, 9,'Systembereich reservi'+
[N52]  162:            'eren (J/N) ? ');
[P96]  163:   WriteXY ( 3,11,'Diskettenlabel eingeb'+
[412]  164:            'en   (J/N) ? ');
[2D6]  165:   WriteXY ( 3,13,'Systemdateien kopiere'+
[7W1]  166:            'n    (J/N) ? ');
[P31]  167:   Repeat
[K82]  168:     GotoXY (4Ø, 3);
[DXØ]  169:     {$I-}
[Y81]  17Ø:       Read (c);
[BXØ]  171:     {$I+}
[T63]  172:     LW := Upcase (c) + ':';
[6T5]  173:   Until (LW[1] >= 'A') And (LW[1] <= 'F');
[P31]  174:   Repeat
[P82]  175:     GotoXY (4Ø, 5);
[DXØ]  176:     {$I-}
[Y81]  177:       Read (c);
[BXØ]  178:     {$I+}
[9U3]  179:   Until (IOResult = Ø) And
[U13]  18Ø:         ((Upcase (c) = 'N') Or
[5S2]  181:         ( Upcase (c) = 'J'));
[PA3]  182:   If Upcase (c) = 'J' Then
[6C4]  183:     Beids := '  ' Else Beids := '/1';
[P31]  184:   Repeat
[U82]  185:     GotoXY (4Ø, 7);
[DXØ]  186:     {$I-}
[Y81]  187:       Read (c);
[BXØ]  188:     {$I+}
[9U3]  189:   Until (IOResult = Ø) And
[9R2]  19Ø:         ((c = '8') or (c = '9'));
[4G4]  191:   If c = '8' Then Sektoren := '/8'
[E93]  192:   Else Sektoren := '  ';
[P31]  193:   Repeat
[Y82]  194:     GotoXY (4Ø, 9);
[DXØ]  195:     {$I-}
[Y81]  196:       Read (c);
[BXØ]  197:     {$I+}
[9U3]  198:   Until (IOResult = Ø) And
[U13]  199:         ((Upcase (c) = 'N') Or
[5S2]  2ØØ:         ( Upcase (c) = 'J'));
[K25]  2Ø1:   If Upcase (c) = 'J' Then Sys := '/B'
[WD2]  2Ø2:   Else Sys := '  ';
[P31]  2Ø3:   Repeat
[Z82]  2Ø4:     GotoXY (4Ø,11);
[DXØ]  2Ø5:     {$I-}
[Y81]  2Ø6:       Read (c);
[BXØ]  2Ø7:     {$I+}
[9U3]  2Ø8:   Until (IOResult = Ø) And
[U13]  2Ø9:         ((Upcase (c) = 'N') Or
[5S2]  21Ø:         ( Upcase (c) = 'J'));
[FU5]  211:   If Upcase (c) = 'J' Then Kennung := '/V'
[933]  212:   Else Kennung := '  ';
[P31]  213:   Repeat
[392]  214:     GotoXY (4Ø,13);
[DXØ]  215:     {$I-}
[Y81]  216:       Read (c);
[BXØ]  217:     {$I+}
[9U3]  218:   Until (IOResult = Ø) And
[U13]  219:         ((Upcase (c) = 'N') Or
[5S2]  22Ø:         ( Upcase (c) = 'J'));
[L25]  221:   If Upcase (c) = 'J' Then Kop := '/S'
[TB2]  222:   Else Kop := '  ';
[9A5]  223:   WriteXY (3,16,'ALLE EINGABEN IN '+
[633]  224:            'ORDNUNG ? < J/N > ');
[P31]  225:   Repeat
[992]  226:     GotoXY (4Ø,16);
[DXØ]  227:     {$I-}
[Y81]  228:       Read (c);
[BXØ]  229:     {$I+}
[9U3]  23Ø:   Until (IOResult = Ø) And
[U13]  231:         ((Upcase (c) = 'N') Or
[5S2]  232:         ( Upcase (c) = 'J'));
[2B3]  233:   If Upcase (c) = 'N' Then
[MUØ]  234:   Begin
[ZTØ]  235:     Ende;
[ØWØ]  236:     Exit;
[ZNØ]  237:   End;
[SU5]  238:   Parameter := Parameter + LW + Beids +
[NE2]  239:                 Sektoren + Sys +
[X92]  24Ø:                 Kennung + Kop;
[KP2]  241:   For a := 1 To 19 Do
[MUØ]  242:   Begin
[624]  243:     If Parameter[a] = ' ' Then
[PR3]  244:       Delete (Parameter,a,1);
[ZNØ]  245:   End;
[V42]  246:   FullWindow;
[DB1]  247:   ClrScr;
[8T5]  248:   Writeln ('Formatiere Diskette in '+
[WV2]  249:           'Laufwerk ',LW,':');
[XF1]  25Ø:   Writeln;
[8T6]  251:   Exec (ExternPfad+'FORMAT.COM',Parameter);
[KA4]  252:  { COM ggf. in EXE ändern ! }
[5P3]  253:   If DosExitCode <> Ø Then
[C23]  254:     Fehlermeldung (1Ø);
[LV1]  255:   CursorOff;
[DB1]  256:   ClrScr;
[ZTØ]  257:   Ende;
[ZNØ]  258: End;
[ØØØ]  259:
[A87]  26Ø: {--- Hauptprogramm Diskette prüfen ------}
[ØØØ]  261:
[4G3]  262: Procedure CheckDisk;
[RNØ]  263: Var
[PG1]  264:   LW : Char;
[MUØ]  265: Begin
[DB1]  266:   ClrScr;
[N44]  267:   xWindow (' D I S K E T T E   '+
[E83]  268:           'P R Ü F E N ',1,1,8Ø,25);
[ER5]  269:   Assign (f,ExternPfad+'CHKDSK.COM');
[KA4]  27Ø:  { COM ggf. in EXE ändern ! }
[DXØ]  271:   {$I-}
[NF1]  272:     Reset (f);
[BXØ]  273:   {$I+}
[5A3]  274:   If IOResult <> Ø Then
[MUØ]  275:   Begin
[P23]  276:     Fehlermeldung (15);
[ØWØ]  277:     Exit;
[ZNØ]  278:   End;
[CE1]  279:   Close (f);
[MN1]  28Ø:   CursorOn;
[P31]  281:   Repeat
[D43]  282:     xWindow ('',3,5,77,8);
[Ø44]  283:    { Laufwerk einlesen   }
[T35]  284:     WriteXY (3,2,' Bitte Laufwerks'+
[UW3]  285:              'buchstaben eingeben ');
[1V1]  286:     ReadLn (LW);
[Ø24]  287:   Until (Upcase (LW) >= 'A') And
[RX2]  288:         (Upcase (LW) < 'G');
[V42]  289:   FullWindow;
[DB1]  29Ø:   ClrScr;
[5V6]  291:   Writeln ('Prüfe Diskette in Laufwerk ',
[W11]  292:            LW,':');
[XF1]  293:   Writeln;
[N56]  294:   Exec (ExternPfad+'CHKDSK.COM',LW+':/f');
[KA4]  295:  { COM ggf. in EXE ändern ! }
[LV1]  296:   CursorOff;
[Y44]  297:   Repeat Until KeyPressed;
[DB1]  298:   ClrScr;
[ZTØ]  299:   Ende;
[ZNØ]  3ØØ: End;
[ØØØ]  3Ø1:
[YØ7]  3Ø2: {--- Hauptprogramm Diskette kopieren   --}
[ØØØ]  3Ø3:
[DE3]  3Ø4: Procedure Diskcopy;
[RNØ]  3Ø5: Var
[SW1]  3Ø6:   c          : Char;
[6X3]  3Ø7:   Parameter : String[5];
[ESØ]  3Ø8:   QLW,
[9D3]  3Ø9:   ZLW        : String[2];
[MUØ]  31Ø: Begin
[DB1]  311:   ClrScr;
[Y35]  312:   xWindow (' D I S K E T T E   K O P I '+
[TG2]  313:           'E R E N ',1,1,8Ø,25);
[ZY5]  314:   Assign (f,ExternPfad+'DISKCOPY.COM');
```

```
[KA4]  315:  { COM ggf. in EXE ändern ! }
[DXØ]  316:   {$I-}
[NF1]  317:     Reset (f);
[BXØ]  318:   {$I+}
[5A3]  319:   If IOResult <> Ø Then
[MUØ]  32Ø:   Begin
[P23]  321:     Fehlermeldung (15);
[ØWØ]  322:     Exit;
[ZNØ]  323:   End;
[CE1]  324:   Close (f);
[ZC2]  325:   Parameter := '';
[MN1]  326:   CursorOn;
[T93]  327:   xWindow ('',1Ø,6,7Ø,2Ø);
[W66]  328:   WriteXY ( 3, 4,'Bitte Quellaufwerks-'+
[M74]  329:            'Buchstaben eingeben   : ');
[X76]  33Ø:   WriteXY ( 3, 7,'Bitte Ziellaufwerks-'+
[M74]  331:            'Buchstaben eingeben   : ');
[P31]  332:   Repeat
[C92]  333:     GotoXY (48, 4);
[DXØ]  334:     {$I-}
[Y81]  335:       Read (c);
[BXØ]  336:     {$I+}
[Ø73]  337:     QLW := Upcase (c) + ':';
[BS3]  338:   Until (QLW[1] >= 'A') And
[XA2]  339:         (QLW[1] <= 'F');
[P31]  34Ø:   Repeat
[K92]  341:     GotoXY (48, 7);
[DXØ]  342:     {$I-}
[Y81]  343:       Read (c);
[BXØ]  344:     {$I+}
[K73]  345:     ZLW := Upcase (c) + ':';
[MS3]  346:   Until (ZLW[1] >= 'A') And
[SB2]  347:         (ZLW[1] <= 'F');
[1N2]  348:   If QLW = ZLW Then
[MUØ]  349:   Begin
[E23]  35Ø:     Fehlermeldung (11);
[ZTØ]  351:     Ende;
[ØWØ]  352:     Exit;
[ZNØ]  353:   End;
[X95]  354:   WriteXY (3,1Ø,'ALLE EINGABEN IN '+
[633]  355:            'ORDNUNG ? < J/N > ');
[P31]  356:   Repeat
[992]  357:     GotoXY (4Ø,16);
[DXØ]  358:     {$I-}
[Y81]  359:       Read (c);
[BXØ]  36Ø:     {$I+}
[9U3]  361:   Until (IOResult = Ø) And
[U13]  362:         ((Upcase (c) = 'N') Or
[5S2]  363:         ( Upcase (c) = 'J'));
[2B3]  364:   If Upcase (c) = 'N' Then
[MUØ]  365:   Begin
[ZTØ]  366:     Ende;
[ØWØ]  367:     Exit;
[ZNØ]  368:   End;
[ØX3]  369:   Parameter := QLW+' '+ZLW;
[V42]  37Ø:   FullWindow;
[DB1]  371:   ClrScr;
[AR5]  372:   Writeln ('Kopiere von ',QLW,' nach ',
[LWØ]  373:            ZLW);
[XF1]  374:   Writeln;
[D65]  375:   Exec (ExternPfad+'DISKCOPY.COM',
[NZ1]  376:       Parameter);
[KA4]  377:  { COM ggf. in EXE ändern ! }
[5P3]  378:   If DosExitCode <> Ø Then
[C23]  379:     Fehlermeldung (1Ø);
[LV1]  38Ø:   CursorOff;
[DB1]  381:   ClrScr;
[ZTØ]  382:   Ende;
[ZNØ]  383: End;
[ØØØ]  384:
[LL6]  385: {--- externe Programme starten ----------}
[ØØØ]  386:
[6S3]  387: Procedure RunProgram;
[6XØ]  388: Const
[AB6]  389:   Leer = '
                                                  ';
[533]  39Ø:  { 64 Leerzeichen }
[RNØ]  391: Var
[LUØ]  392:   Path,
[754]  393:   Parameter : String[64];
[4Ø2]  394:   a         : Byte;
[MUØ]  395: Begin
[Ø22]  396:   Path := Leer;
[DB1]  397:   ClrScr;
[Y55]  398:   xWindow (' E X T E R N E S   P R O G '+
[N53]  399:            'R A M M   S T A R T E N ',
[9C1]  4ØØ:            1,1,8Ø,25);
[WE1]  4Ø1:   AktPath;
[MN1]  4Ø2:   CursorOn;
[943]  4Ø3:   xWindow ('',3,4,77,8);
[C46]  4Ø4:   WriteXY ( 3, 2,'Bitte Programmnamen '+
[K72]  4Ø5:            'eingeben : ');
[L72]  4Ø6:   GotoXY ( 3, 3);
[372]  4Ø7:   ReadLn (Path);
[C33]  4Ø8:   a := FindFile (Path);
[S91]  4Ø9:   Case a Of
[W81]  41Ø:     Ø : Begin
[U23]  411:           Fehlermeldung (17);
[ZTØ]  412:           Ende;
[ØWØ]  413:           Exit;
[ZNØ]  414:         End;
[RP3]  415:     1 : Path := Path + '.COM';
[VR3]  416:     2 : Path := Path + '.EXE';
[7N3]  417:     3 : Path := Path + '.BAT';
[WP1]  418:     Else Begin
[S23]  419:            Fehlermeldung (16);
[ZTØ]  42Ø:            Ende;
[ØWØ]  421:            Exit;
[ZNØ]  422:          End;
[ZNØ]  423:   End;
[5B3]  424:   xWindow ('',3,1Ø,77,14);
[KW6]  425:   WriteXY ( 3, 2,'Bitte Programmparameter '
       +
[A53]  426:            'eingeben oder CR !');
[L72]  427:   GotoXY ( 3, 3);
[ZC2]  428:   Parameter := '';
[U63]  429:   ReadLn (Parameter);
[7V4]  43Ø:   Parameter := Parameter + Leer;
[V42]  431:   FullWindow;
[DB1]  432:   ClrScr;
[UK3]  433:   Exec (Path,Parameter);
[LV1]  434:   CursorOff;
[DB1]  435:   ClrScr;
[ZTØ]  436:   Ende;
[ZNØ]  437: End;
[ØØØ]  438:
[MUØ]  439: Begin
[PK2]  44Ø:   ExternPfad := '';
[KNØ]  441: End.
```

Listing 4. Die Unit »extern.pas« wickelt menügesteuerte Aufrufe externer MS-DOS-Funktionen ab

```
[PD5]    1: { Programmname: EASYDOS.PAS  }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[CP4]    5: {$R-,S+,I+,D+,T-,F-,V+,B-,N-,L+ }
[M42]    6: {$M 4ØØØ,Ø,1ØØØ}
[ØØØ]    7:
[HD4]    8: { E A S Y D O S   Version 1.1Ø
[X24]    9:   Autor: Karlheinz Büker
[4L3]   1Ø:          Moritz-Kabaker-Str. 53
[8P2]   11:          492Ø Lemgo              }
[ØØØ]   12:
[2Y2]   13: Program EasyDos;
[ØØØ]   14:
[ASØ]   15: Uses
[PB6]   16:   Dos, Crt, Global, Extern, General, Disk;
[ØØØ]   17:
[6XØ]   18: Const
[H84]   19:   PrgExit : Boolean = False;
[ØØØ]   2Ø:
[RNØ]   21: Var
[Ø62]   22:   ShowTimeVec,
[1K3]   23:   OldExit      : Pointer;
[PG2]   24:   SKZ          : Char;
[CB1]   25:   Eingabe,
[88Ø]   26:   b,
[N52]   27:   d            : Byte;
[YPØ]   28:   Std,
[TPØ]   29:   Min,
[DPØ]   3Ø:   Sek,
[6K2]   31:   Sek1ØØ       : Word;
[BA1]   32:   Stunde,
[991]   33:   Minute,
[UW3]   34:   Sekunde      : String [2];
[ØØØ]   35:
[8V6]   36: {--- Exit-Prozedur (Nur zur Sicherheit);
[2G5]   37:      setzt Ticker-Int-Vektor zurück
[PH6]   38:      und schließt evtl. offene Dateien --}
[ØØØ]   39:
[ZWØ]   4Ø: {$F+}
[R73]   41: Procedure NewExit;
[1XØ]   42: {$F-}
[MUØ]   43: Begin
[DXØ]   44:   {$I-}
[CE1]   45:     Close (f);
[UE1]   46:     Close (t);
[BXØ]   47:   {$I+}
[KN4]   48:   SetIntVec (28,ShowTimeVec);
[EG3]   49:   ExitProc := OldExit;
[ZNØ]   5Ø: End;
[ØØØ]   51:
[KP3]   52: {--- Ticker-Interrupt
[GN5]   53:      gibt Uhrzeit und Datum aus ---------}
```

Listing 5. Das Hauptprogramm der Benutzeroberfläche: »easydos.pas«

```
[ØØØ]   54:
[ZWØ]   55: {$F+}
[9R2]   56: Procedure Time;
[8X1]   57: Interrupt;
[1XØ]   58: {$F-}
[MUØ]   59: Begin
[DR4]   6Ø:   GetTime (Std,Min,Sek,Sek1ØØ);
[A82]   61:   GotoXY ( 4,24);
[SZ2]   62:   Str (Std,Stunde);
[XX2]   63:   Str (Min,Minute);
[X24]   64:   If Length(Minute) = 1 Then
[B63]   65:     Minute := 'Ø'+Minute;
[E23]   66:   Str (Sek,Sekunde);
[Y54]   67:   If Length(Sekunde) = 1 Then
[8M3]   68:     Sekunde := 'Ø'+Sekunde;
[LD5]   69:   Write ('▌',Stunde:2,':',Minute,':',
[BC2]   7Ø:           Sekunde,'▐');
[2R4]   71:   GetDate (Std,Min,Sek,Sek1ØØ);
[W92]   72:   GotoXY (66,24);
[M95]   73:   Write ('▌',Sek:2,'.',Min:2,'.',Std:2,
[DZØ]   74:         '▐');
[ZNØ]   75: End;
[ØØØ]   76:
[5H5]   77: {--- Zeigt Menüpunkte und fragt
[CY4]   78:      Funktion ab ------------------------}
[ØØØ]   79:
[LR3]   8Ø: Procedure ZeigeMenue;
[MUØ]   81: Begin
[S64]   82:   TextColors (White,Black);
[LV1]   83:   CursorOff;
[DB1]   84:   ClrScr;
[RY4]   85:   xWindow(' E A S Y D O S   1.1Ø  -  '+
[BM2]   86:           'M E N Ü ',1,1,8Ø,25);
[6F4]   87:   TextColors (Black,Yellow);
[8E5]   88:   WriteXY ( 2, 2, ' Taste         ALLEIN'+
[YV2]   89:                '              +CTRL    '+
[SX2]   9Ø:                '           +SHIFT      '+
[2RØ]   91:                '    ');
[BP2]   92:   For b := 1 To 1Ø Do
[MUØ]   93:   Begin
[BT2]   94:     GotoXY ( 2,2 * b + 2);
[WL2]   95:     Write ('  F',b,'  ');
[ZNØ]   96:   End;
[HE4]   97:   TextColors (Yellow,Green);
[6L6]   98:   WriteXY ( 9, 4,' System-Informationen ');
[KA1]   99:  {       F1 }
[5D6]  1ØØ:   WriteXY ( 9, 6,' DOS-Version anzeigen ');
[NA1]  1Ø1:  {       F2 }
[8H6]  1Ø2:   WriteXY ( 9, 8,' Verify-Status setzen ');
[SA1]  1Ø3:  {       F3 }
[3Z5]  1Ø4:   WriteXY ( 9,1Ø,' Disk formatieren     ');
[VA1]  1Ø5:  {       F4 }
[5P5]  1Ø6:   WriteXY ( 9,12,' Disk kopieren        ');
[YA1]  1Ø7:  {       F5 }
[NM5]  1Ø8:   WriteXY ( 9,14,' Disk Prüfen          ');
[1B1]  1Ø9:  {       F6 }
[746]  11Ø:   WriteXY ( 9,16,' Programm starten     ');
[4B1]  111:  {       F7 }
[BD6]  112:   WriteXY ( 9,18,' Systemzeit setzen    ');
[7B1]  113:  {       F8 }
[KG6]  114:   WriteXY ( 9,2Ø,' Systemdatum setzen   ');
[AB1]  115:  {       F9 }
[7E6]  116:   WriteXY ( 9,22,' Drucker-Einstellung  ');
[KC1]  117:  {       F1Ø}
[1A6]  118:   WriteXY (33, 4,' Directory anzeigen   ');
[3T1]  119:  {  CTRL F1 }
[FK6]  12Ø:   WriteXY (33, 6,' Directory durchsuchen');
[6T1]  121:  {  CTRL F2 }
[L96]  122:   WriteXY (33, 8,' Directory wechseln   ');
[9T1]  123:  {  CTRL F3 }
[A46]  124:   WriteXY (33,1Ø,' Directory anlegen    ');
[CT1]  125:  {  CTRL F4 }
[Z96]  126:   WriteXY (33,12,' Directory löschen    ');
[FT1]  127:  {  CTRL F5 }
[FB6]  128:   WriteXY (33,14,' verstecktes Dir. anl.');
[KT1]  129:  {  CTRL F6 }
[3P6]  13Ø:   WriteXY (33,16,' Verzeichnisstruktur  ');
[NT1]  131:  {  CTRL F7 }
[R86]  132:   WriteXY (33,18,' DOS-Kommandozeile    ');
[ST1]  133:  {  CTRL F8 }
[RC6]  134:   WriteXY (33,2Ø,' Extern-Pfad setzen   ');
[VT1]  135:  {  CTRL F9 }
[DW5]  136:   WriteXY (33,22,' Zum DOS-Modus        ');
[3V1]  137:  {  CTRL F1Ø}
[3A6]  138:   WriteXY (57, 4,' Dateinamen ändern    ');
[UY1]  139:  { SHIFT F1 }
[5T5]  14Ø:   WriteXY (57, 6,' Datei löschen        ');
[XY1]  141:  { SHIFT F2 }
[T86]  142:   WriteXY (57, 8,' Textdatei anzeigen   ');
[ØZ1]  143:  { SHIFT F3 }
[3V5]  144:   WriteXY (57,1Ø,' Datei kopieren       ');
[3Z1]  145:  { SHIFT F4 }
[5P6]  146:   WriteXY (57,12,' Dateiattribute setzen');
[6Z1]  147:  { SHIFT F5 }
[W26]  148:   WriteXY (57,14,' Datei verschieben    ');
[9Z1]  149:  { SHIFT F6 }
[V26]  15Ø:   WriteXY (57,16,' Dateizeit ändern     ');
[CZ1]  151:  { SHIFT F7 }
[VB6]  152:   WriteXY (57,18,' Dateieintrag suchen  ');
[FZ1]  153:  { SHIFT F8 }
[AN5]  154:   WriteXY (57,2Ø,' Datei drucken        ');
[KZ1]  155:  { SHIFT F9 }
[HX5]  156:   WriteXY (57,22,' EASYDOS beenden      ');
[UØ2]  157:  { SHIFT F1Ø}
[S64]  158:   TextColors (White,Black);
[LK6]  159:   WriteXY (3Ø,25,'Bitte Funktion wählen!');
[MN7]  16Ø:   WriteXY ( 4,23,'▌          ▐');
[VN7]  161:   WriteXY ( 4,25,'▌          ▐');
[XW8]  162:   WriteXY (66,23,'▌          ▐');
[2X8]  163:   WriteXY (66,25,'▌          ▐');
[UM4]  164:   GetIntVec (28,ShowTimeVec);
[F16]  165:  {   Ticker-Interruptvektor sichern }
[FB3]  166:   SetIntVec (28,@Time);
[AW5]  167:  { ... und auf Time-Prozedur setzen }
[P31]  168:   Repeat
[NH2]  169:     SKZ := ReadKey;
[TW2]  17Ø:     If SKZ = Chr(Ø) Then
[MUØ]  171:     Begin
[2G4]  172:     SetIntVec(28,ShowTimeVec);
[UY5]  173:    { Ticker-Interrupt auf alten Wert  }
[2Z1]  174:       c := ReadKey;
[YN1]  175:       d := Ord(c);
[1A1]  176:       Case d Of
[RM4]  177:         59 : SysInfo;          { F1          }
[N95]  178:         6Ø : GetDosVersion;    { F2          }
[XH5]  179:         61 : SetVerifyStatus; { F3          }
[7Y4]  18Ø:         62 : FormatDisk;       { F4          }
[VN4]  181:         63 : DiskCopy;         { F5          }
[HS4]  182:         64 : CheckDisk;        { F6          }
[XØ5]  183:         65 : RunProgram;       { F7          }
[DC5]  184:         66 : SetSystemTime;    { F8          }
[TA5]  185:         67 : SetSystemDate;    { F9          }
[WE5]  186:         68 : InstallPrinter;   { F1Ø         }
[235]  187:         94 : ShowDir;          { CTRL F1     }
[EA5]  188:         95 : SearchDir;        { CTRL F2     }
[995]  189:         96 : ChangeDir;        { CTRL F3     }
[115]  19Ø:         97 : MakeDir;          { CTRL F4     }
[PA5]  191:         98 : RemakeDir;        { CTRL F5     }
[YP5]  192:         99 : MakeHiddenDir;    { CTRL F6     }
[EU4]  193:        1ØØ : ShowTree;         { CTRL F7     }
[VV4]  194:        1Ø1 : DosCommand;       { CTRL F8     }
[RB5]  195:        1Ø2 : SetExternPath;    { CTRL F9     }
[GY4]  196:        1Ø3 : RunCommand;       { CTRL F1Ø    }
[ZC5]  197:         84 : RenameFile;       { SHIFT F1    }
[MC5]  198:         85 : DeleteFile;       { SHIFT F2    }
[U75]  199:         86 : ShowFile;         { SHIFT F3    }
[F75]  2ØØ:         87 : CopyFile;         { SHIFT F4    }
[665]  2Ø1:         88 : SetAttr;          { SHIFT F5    }
[L75]  2Ø2:         89 : MoveFile;         { SHIFT F6    }
[AH5]  2Ø3:         9Ø : SetFileTime;      { SHIFT F7    }
[5D5]  2Ø4:         91 : SearchFile;       { SHIFT F8    }
[GD5]  2Ø5:         92 : PrintFile;        { SHIFT F9    }
[PW5]  2Ø6:         93 : PrgExit := True; { SHIFT F1Ø }
[ZNØ]  2Ø7:       End;
[ZNØ]  2Ø8:     End;
[T33]  2Ø9:   Until SKZ = Chr(Ø);
[ZNØ]  21Ø: End;
[ØØØ]  211:
[R95]  212: {--- H A U P T P R O G R A M M ----------}
[ØØØ]  213:
[MUØ]  214: Begin
[5E3]  215:   OldExit := ExitProc;
[PL3]  216:   ExitProc := @NewExit;
[RB7]  217:  { Zur Sicherheit Exit-Prozedur einbinden }
[E63]  218:   CheckBreak := False;
[163]  219:   CheckSnow := False;
[AE2]  22Ø:   TextMode (C8Ø);
[S64]  221:   TextColors (White,Black);
[DB1]  222:   ClrScr;
[H21]  223:   Start;
[P31]  224:   Repeat
[612]  225:     ZeigeMenue;
[PG2]  226:   Until PrgExit;
[W43]  227:   CheckBreak := True;
[MN1]  228:   CursorOn;
[DB1]  229:   ClrScr;
[KNØ]  23Ø: End.
```

Listing 5. Das Hauptprogramm der Benutzeroberfläche: »easydos.pas«

Komprimieren Sie Ihre Programme

Sicher haben Sie auch schon von der Möglichkeit gehört, beliebige Dateien mit Hilfe von »Packern« zu komprimieren und bei Bedarf wieder in die ursprüngliche Länge umzuwandeln. Das hier vorgestellte Utility arbeitet nach einem anderen Prinzip: Die komprimierten Programme sind auch ohne umständliches Entpacken lauffähig.

Ein mit dem Utility »pack.exe« bearbeitetes Programm benötigt weniger Platz und kann dennoch jederzeit gestartet werden. Das Entpacken geschieht beim Start des komprimierten Programms automatisch, da der zuständige Algorithmus gleich mit im Programmcode enthalten ist.
Weil durch die Komprimierung alle Texte und Programmroutinen nicht mehr zu entschlüsseln sind, eignet sich »pack« auch zum Codieren. So ist zum Beispiel das Disassemblieren und nachträgliche Ändern eines komprimierten Programms nicht mehr möglich.
Der wichtigste Punkt, der über die Qualität eines Packprogramms entscheidet, ist natürlich die Effektivität des Algorithmus, also die Größe des gepackten Programms. Ein sehr einfacher Packalgorithmus basiert zum Beispiel auf der Wiederholung von Bytes. Anstatt 1000mal das Null-Byte zu speichern, sichert ein solcher Algorithmus nur ein Null-Byte und die Zahl 1000, was in diesem Fall natürlich einen enormen Speicherplatzgewinn bedeutet. Dieses Verfahren verwendet zum Beispiel der Microsoft-Linker bei Angabe der Option »/e« (»exepack«).

Ein ausgeklügelter Pack-Algorithmus verkürzt Ihre Programme auf ein Minimum

Sollten im Originalprogramm dagegen kaum Byte-Wiederholungen vorkommen, so ist dieses Packverfahren nutzlos. Dann eignet sich der hier vorgestellte Algorithmus, der auf Bitebene abläuft, weitaus besser.
Er weist jedem Byte eine festgelegte Bit-Kombination zu, deren Länge zwischen 4 und 10 Bit variiert (siehe Tabelle). Die Zuordnung richtet sich nach der Häufigkeit eines Bytes. Kommt zum Beispiel der Wert 18hex im Originalprogramm am häufigsten vor, so wird er als eine 4-Bit-Kombination codiert. Alle weiteren Werte erhalten je nach der Häufigkeit ihres Vorkommens eine bis zu zehn Bit lange Kombination. Das Resultat ist eine durchschnittliche Verkürzung der Programme um zehn bis zwanzig Prozent, manchmal sogar bis zu vierzig Prozent.
Das Komprimierungsprogramm besteht aus »pack.exe« und »pack2.com«. Beide sind einmal als Hexdump zum schnelleren Abtippen vorhanden (Listings 1 und 2) und einmal als Quelltext für den MASM- oder TASM-Assembler (Listings 3 und 4). »Pack2.com« enthält die Entpack-Routine und wird zu der komprimierten Datei automatisch hinzugefügt.
Komprimierbar sind prinzipiell alle Programme mit der Endung »com«. Dateien mit der Endung »exe« müssen folgende Anforderungen erfüllen: Sie dürfen nicht länger als 64 KByte sein und müssen unter Verwendung des Parameters »/e« mit dem Microsoft-Linker gebunden worden sein (dadurch ist gewährleistet, daß das Programm keine Relokationen enthält). Exe-Programme sollten Sie also nur dann mit »pack« bearbeiten, wenn Sie sicher sind, daß diese Bedingung eingehalten wurde. Die Einschränkung der Länge ist notwendig, da »pack« das gepackte Programm als Com-Datei speichert (die maximale Größe einer Com-Datei ist 64 KByte).
Der Aufruf des Programms geschieht mit der folgenden Syntax:

```
pack <Orginalprogrammname> <Zieldateiname>
```

Das Programm »pack.exe« läuft jetzt folgendermaßen ab: Beide Programmnamen werden gelesen und gesichert, das Originalprogramm, die Zieldatei und »pack2.com« werden zum Lesen beziehungsweise zum Schreiben geöffnet. Anschließend erfolgt eine Prüfung des Originalprogramms, ob es den obengenannten Anforderungen entspricht.
Sind diese Schritte erfolgreich ausgeführt, wird »pack2.com« als erstes in die Zieldatei geschrieben. Nun liest das Utility das Originalprogramm komplett und legt eine Byte-Häufigkeitstabelle an. Daraus entsteht die Decodiertabelle, welche die Bit-Kombination jedes Bytes enthält. Mit Hilfe dieser Tabelle codiert die Routine dann das Originalprogramm und überträgt es in die Zieldatei. Zum Schluß werden noch einige Laufvariablen eingefügt, wie zum Beispiel die Größe des Originalprogramms und eine Prüfsumme. »Pack« schließt alle Dateien, und der Kompressionsvorgang ist beendet.
Beim Starten eines gepackten Programms verschiebt die Routine »pack2.com« das gesamte Programm in ein anderes Segment. Dort wird der codierte Teil entpackt und im Ausgangssegment gespeichert. Ist der codierte Teil fehlerhaft, so meldet sich das Programm mit

```
Checksummen-Fehler!
```

Besitzer des MASM- oder TASM-Assemblers verwenden zum Abtippen die Listings 3 und 4. Die assemblierte und gelinkte Datei aus Listing 4 muß zusätzlich noch mit »exe2bin« in eine Com-Datei umgewandelt werden. Als Alternative dazu können Sie auch die Hexdumps in Listing 1 und 2 mit einem Texteditor eingeben. Das Programm »SCheck« (das erste Listing in diesem Heft) wandelt sie dann in lauffähige Programme um.

(Richard Lesser/tr)

Name	pack.asm und pack2.asm
Sprache	MASM ab 4.0, TASM
Grafikkarte	jede
Besonderheiten	keine

Typ-Bits	Erkennungs-Bits	Gesamt	Darstellende Bytes
000	x	4 Bit	2 Byte
001	x	4 Bit	2 Byte
010	xx	5 Bit	4 Byte
011	xxx	6 Bit	8 Byte
100	xxxx	7 Bit	16 Byte
101	xxxxx	8 Bit	32 Byte
110	xxxxxx	9 Bit	64 Byte
111	xxxxxxx	10 Bit	128 Byte
			256 Byte

Tabelle. Der Aufbau der Bitkombinationen: Insgesamt können alle 256 verschiedenen Bytes dargestellt werden

```
[GK5]     1: 4D 5A 74 01 05 00 00 00 20 00 3C 0E FF FF AC 0E
[K25]     2: 80 00 00 00 10 00 64 00 1E 00 00 00 01 00 00 00
[0Z4]     3: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]     4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]     5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]     6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]     7: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]     8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]     9: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    11: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    12: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    13: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    14: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    15: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    16: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    17: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    18: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    19: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    21: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    22: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    23: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    24: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    25: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    26: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    27: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    28: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    29: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    31: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]    32: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0E5]    33: 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41
[VD5]    34: 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53
[DE5]    35: 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B
[0E5]    36: 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41
[VD5]    37: 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53
[DE5]    38: 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B
[0E5]    39: 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41
[VD5]    40: 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53
[DE5]    41: 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B
[0E5]    42: 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41
[VD5]    43: 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53
[DE5]    44: 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B
[0E5]    45: 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41
[VD5]    46: 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53
[DE5]    47: 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B
[0E5]    48: 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41
[VD5]    49: 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53
[DE5]    50: 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B
[0E5]    51: 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41
[VD5]    52: 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53
[DE5]    53: 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B
[0E5]    54: 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41
[VD5]    55: 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B 2E 53
[DE5]    56: 54 41 43 4B 2E 53 54 41 43 4B 2E 53 54 41 43 4B
[SE5]    57: 00 40 00 B1 50 41 43 4B 32 2E 43 4F 4D 09 00 B2
[SD5]    58: 00 0C E3 B0 01 01 00 B2 00 0C 00 B0 0A 44 69 65
[GD5]    59: 73 65 73 20 50 72 6F 67 72 61 6D 6D 20 6D 61 63
[3D5]    60: 68 74 20 61 75 73 20 43 4F 4D 20 75 6E 64 20 45
[ZE5]    61: 58 45 20 50 72 6F 67 72 61 6D 6D 6E 0A 0D 65
[5F5]    62: 69 6E 20 67 65 70 61 63 6B 74 65 73 20 43 4F 4D
[PN5]    63: 20 50 72 6F 67 72 61 6D 6D 2E 0A 0D 0A 53 79 6E
[5E5]    64: 74 61 78 3A 20 43 20 66 69 6C 65 31 20 66 69 6C
[EK5]    65: 65 32 20 2D 3E 20 66 69 6C 65 31 3D 49 6E 70 75
[8F5]    66: 74 2D 66 69 6C 65 20 20 66 69 6C 65 32 3D 4F 75
[1F5]    67: 74 70 75 74 2D 66 69 6C 65 0A 0D 24 44 61 74 65
[3E5]    68: 69 20 46 65 68 6C 65 72 21 0A 0D 24 45 78 65 2D
[PD5]    69: 66 69 6C 65 20 7A 75 20 6C 61 6E 67 21 0A 0D 24
[7D5]    70: 45 73 20 64 81 72 66 65 6E 20 6B 65 69 6E 65 20
[6K5]    71: 45 78 65 2D 66 69 6C 65 20 52 65 6C 6F 6B 61 74
[9L5]    72: 69 6F 6E 65 6E 20 76 6F 72 68 61 6E 64 65 6E 20
[BE5]    73: 73 65 69 6E 21 0A 0D 50 72 6F 67 72 61 6D 6D 20
[GL5]    74: 6D 69 74 20 2F 65 20 4F 70 74 69 6F 6E 20 6E 65
[NH5]    75: 75 20 6C 69 6E 6B 65 6E 2E 0A 0D 24 10 01 B2 00
[4S5]    76: 0E 00 B0 B8 18 00 8E C0 FC BE 80 00 AC 32 E4 0A
[MM5]    77: C0 75 0F 06 1F B4 09 8D 16 62 E3 CD 21 B8 01 4C
[RT5]    78: CD 21 8B C8 8D 3E 00 00 AC 3C 20 74 FB 4E AC 3C
[YD5]    79: 20 75 0C 81 FF 20 00 77 21 8D 3E 20 00 EB 19 3C
[AH5]    80: 0D 74 15 3C 60 72 02 24 DF 3C 21 72 C6 3C 5C 77
[0R5]    81: C2 81 FF 40 00 74 BC AA E2 D4 81 FF 20 00 76 B3
[CK5]    82: 06 1F E8 0B 00 E8 42 00 E8 F7 02 B8 00 4C CD 21
[EG5]    83: B8 00 3D 8D 16 00 00 CD 21 A3 4A 00 73 0D 8D 16
[5N5]    84: F2 E3 B4 09 CD 21 B8 02 4C CD 21 B4 3C 8D 16 20
[3K5]    85: 00 B9 20 00 CD 21 A3 4C 00 72 E3 B8 00 3D 8D 16
[RK5]    86: 40 00 CD 21 A3 4E 00 72 D5 C3 B4 3F 8B 1E 4A 00
[VH5]    87: B9 00 01 8D 16 50 70 CD 21 72 49 B8 00 42 33 C9
[SN5]    88: 33 D2 CD 21 72 3E B4 3F 8B 1E 4E 00 B9 00 70 8D
[YM5]    89: 16 50 00 CD 21 72 2D A3 50 E3 8D 3E 53 00 B8 FE
[NR5]    90: FF AB 33 C0 AB B8 00 01 AB 33 C0 AB 8D 36 50 70
[5P5]    91: AD 3D 4D 5A 75 5D AD AD 3D 83 00 72 15 8D 16 02
[K06]    92: E4 E9 7E FF 8D 16 F2 E3 E9 77 FF 8D 16 16 E4 E9
[XY5]    93: 70 FF AD 0B C0 75 F4 AD A3 5A E3 AD 3D FF 0F 77
[HR5]    94: DC AD AD 05 10 00 50 8D 3E 53 00 A5 58 AB AD A5
[KK5]    95: AD 05 10 00 AB B4 3F 8B 1E 4A 00 B9 00 01 8D 16
[NM5]    96: 50 70 CD 21 72 BE A1 5A E3 2D 10 00 A3 5A E3 0B
[HF5]    97: C0 75 E2 B4 40 8D 16 50 00 8B 1E 4C 00 8B 0E 50
[YF5]    98: E3 CD 21 72 9F C7 06 50 E3 00 00 B8 01 42 33 C9
[EK5]    99: 33 D2 8B 1E 4A 00 CD 21 72 18 A3 56 E3 89 16 58
[UH5]   100: E3 B4 3F 8B 1E 4A 00 B9 00 70 8D 16 50 00 CD 21
[XH5]   101: 73 03 E9 6F FF 01 06 50 E3 E8 61 00 3D 00 70 74
[1L5]   102: E0 B8 00 42 8B 16 56 E3 8B 0E 58 E3 8B 1E 4A 00
[5N5]   103: CD 21 72 DE E8 68 00 33 ED 8D 06 50 70 A3 5E E3
[XF5]   104: E8 B0 00 B4 3F 8B 1E 4A 00 B9 00 70 8D 16 50 00
[2R5]   105: CD 21 72 BE E8 B0 00 3D 00 70 74 E7 E8 4D 01 B8
[4R5]   106: 00 42 33 C9 BA 0B 00 8B 1E 4C 00 CD 21 72 A3 8D
[XE5]   107: 16 50 E3 B4 40 B9 06 00 CD 21 72 96 C3 50 53 51
[DL5]   108: 52 56 57 8B C8 8D 36 50 00 AC 32 E4 D1 E0 8B D8
[8U5]   109: 81 C3 50 E0 FF 07 E2 F1 5F 5E 5A 59 5B 58 C3 B9
[0H5]   110: 02 00 BD 00 F8 8D 3E 50 E2 51 57 E8 22 00 8B C5
[PW5]   111: AB 8B C7 5F 2D 52 E0 D1 E8 AA 8B C5 45 E2 EB 32
[MY5]   112: C0 FE C4 8B E8 59 80 FC F9 74 DE D0 E1 73 DA C3
[TM5]   113: 51 33 DB 8D 36 50 E0 B9 00 01 AD 50 80 E4 F8 80
[NW5]   114: FC F8 58 74 08 3B C3 72 04 8B D8 8B FE E2 EB 4F
[YT5]   115: 4F 59 C3 51 B9 80 00 8D 36 50 E2 8B 3E 5E E3 F3
[8L5]   116: A5 89 3E 5E E3 59 C3 50 53 51 52 55 8B C8 8D 36
[CU5]   117: 50 00 8B 2E 5C E3 8B 3E 5E E3 A1 60 E3 8B D0 AC
[GK5]   118: 51 56 32 E4 01 06 52 E3 D1 E0 05 50 E0 8B F0 AD
[RT5]   119: 92 52 D0 EE 9C D0 EE 9C D0 EE 9C B1 03 E8 2F 00
[2T5]   120: 5A 80 E6 07 0A F6 75 02 FE C6 8A CE 32 ED 8B F1
[9V5]   121: D0 EA 9C E2 FB 8B CE E8 15 00 5E 59 E2 BF 89 2E
[GT5]   122: 5C E3 89 3E 5E E3 A3 60 E3 5D 5A 59 5B 58 C3 5A
[LT5]   123: 9D D0 D0 45 83 FD 08 72 2D 33 ED AA FF 06 54 E3
[FK5]   124: 32 C0 81 FF 50 E0 75 1E 50 53 51 52 B4 40 8B 1E
[FL5]   125: 4C 00 B9 00 70 8D 16 50 70 8B FA CD 21 73 03 E9
[XM5]   126: E2 FD 5A 59 5B 58 FE C9 75 C6 52 C3 50 53 51 52
[RP5]   127: 57 A1 60 E3 8B 0E 5C E3 8B 3E 5E E3 80 E9 08 F6
[6X5]   128: D9 D3 E0 AA FF 06 54 E3 B4 40 8B 1E 4C 00 8B CF
[7M5]   129: 81 E9 4F 70 8D 16 50 70 CD 21 72 C3 5F 5A 59 5B
[TR5]   130: 58 C3 B4 3E 8B 1E 4A 00 CD 21 B4 3E 8B 1E 4C 00
[XR5]   131: CD 21 B4 3E 8B 1E 4E 00 CD 21 C3 78 03 B2 C3 08
[2H6]   132: 00 B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
[G75]   133: 00 00 60 0E 00 00 34 01 80 01 00 00 98 0E 52 42
[NC5]   134: 8C C0 05 10 00 0E 1F A3 04 00 03 06 0C 00 8E C0
[1N5]   135: 8B 0E 06 00 8B F9 4F 8B F7 FD F3 A4 50 B8 32 00
[CV5]   136: 50 CB 8C C3 8C D8 48 8E D8 8E C0 BF 0F 00 B9 10
[ES5]   137: 00 B0 FF F3 AE 47 8B F7 8B C3 48 8E C0 BF 0F 00
[BW5]   138: B1 04 8B C6 F7 D0 D3 E8 74 0A 8C DA 2B D0 8E DA
[MW5]   139: 81 CE F0 FF 8B C7 F7 D0 D3 E8 74 0A 8C C2 2B D0
[7Z5]   140: 8E C2 81 CF F0 FF AC 8A D0 4E AD 8B C8 46 8A C2
[0N5]   141: 24 FE 3C B0 75 06 AC F3 AA EB 07 90 3C B2 75 56
[8N5]   142: F3 A4 8A C2 A8 01 74 B8 BE 12 01 0E 1F 8B 1E 04
[DW5]   143: 00 FC 33 D2 AD 8B C8 E3 0E 8B C2 03 C3 8E C0 AD
[RL5]   144: 8B F8 26 01 1D E2 F8 81 FA 00 F0 74 06 81 C2 00
[GK5]   145: 10 EB E1 8B C3 8B 3E 08 00 8B 36 0A 00 03 F0 01
[WK5]   146: 06 02 00 2D 10 00 8E D8 8E C0 BB 00 00 FA 8E D6
[XV5]   147: 8B E7 FB 2E FF 2F B4 40 BB 02 00 B9 16 00 8C CA
[FV5]   148: 8E DA BA FC 00 CD 21 B8 FF 4C CD 21 50 61 63 6B
[BD5]   149: 65 64 20 66 69 6C 65 20 69 73 20 63 6F 72 72 75
[215]   150: 70 74 01 00 01 E6 00 00 00 00 00 00 00 00 00 00
[0Z4]   151: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[071]   152: 00 00 00 00
```

Listing 1. Das Hexdump von »pack.exe«

```
[135]     1: E9 FC 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[EB5]     2: 00 28 43 29 20 52 2E 4C 65 73 73 65 72 20 31 39
[PK5]     3: 38 38 00 1E 06 1F 8D 36 51 02 2B FF B9 80 00 F3
[SP5]     4: A5 07 5D BF 00 01 FA 89 26 E6 01 AD 86 E0 B2 08
[MW5]     5: EB 1A B2 08 AC EB 1F B2 08 AC EB 22 B2 08 AC EB
[DU5]     6: 25 B2 08 AC EB 3B 2A C0 FE C1 EB 23 2A C9 D1 E0
[4Z5]     7: D0 D1 FE CA 74 DC D1 E0 D0 D1 FE CA 74 D9 D1 E0
[4T5]     8: D0 D1 FE CA 74 D6 8B E0 B8 01 00 0A C9 74 D7 D3
[5Y5]     9: E0 8B D8 8B C4 8A E9 2A C9 D1 E0 D0 D1 FE CA 74
[6S5]    10: C0 FE CD 75 F4 03 D9 8A 0F 29 0E 0D 01 26 88 0D
[0L5]    11: 47 4D 75 B8 8B 26 E6 01 FB A1 0D 01 0B C0 74 06
[AU5]    12: 8D 16 EA 01 EB 6A 06 1F 58 8C DB 03 C3 2E A3 E8
[GN5]    13: 01 58 2E A3 E6 01 5B 59 58 8C DA 03 DA FA 8B E1
[UT5]    14: 8E D3 FB 2E FF 2E E6 01 4F 55 54 20 4F 46 20 4D
[6D5]    15: 45 4D 4F 52 59 21 0A 0D 07 24 43 48 45 43 4B 53
[9K5]    16: 55 4D 4D 45 4E 2D 46 45 48 4C 45 52 21 0A 24 FC
[WT5]    17: BE 03 01 B9 06 00 50 AD E2 FC B4 4A BB 00 10 CD
[CH5]    18: 21 72 09 B4 48 BB 20 10 CD 21 73 0D 8D 16 D8 01
[TM5]    19: B4 09 CD 21 B8 01 4C CD 21 A3 E8 01 C7 06 E6 01
[TR5]    20: 23 01 8E C0 B4 49 CD 21 72 E2 FC BE 00 01 8B FE
[WM5]    21: B9 51 01 03 0E 0F 01 41 D1 E9 F3 A5 2E FF 2E E6
[L70]    22: 01
```

Listing 2. »pack2.com« als Hexdump

```
[Ø83]    1: title Packprogramm
[5F1]    2: name pack
[ØØØ]    3:
[YV3]    4: ;Autor: Richard Lesser
[3P4]    5: ;Copyright Everts & Hagedorn
[3E4]    6: ;DOS International Extra 7
[ØØØ]    7:
[HW3]    8: ;Assemblieren & Linken:
[UU1]    9: ; masm pack;
[T92]   1Ø: ; link /e pack;
[ØØØ]   11:
[XG1]   12:              .radix 1Øh
[D46]   13: stackseg     segment para stack 'stack'
[ØS2]   14:              db 4Ø dup ('.STACK')
[AF2]   15: stackseg     ends
[ØØØ]   16:
[F33]   17: dataseg      segment
[4X2]   18: file1        db 2Ø dup(Ø)
[K17]   19: ;Speicher für Namen des Orginalprogramms
[6X2]   2Ø: file2        db 2Ø dup(Ø)
[8P5]   21: ;Speicher für Namen der Zieldatei
[9L3]   22: file3        db 'PACK2.COM',Ø
[5P6]   23: ;Pfad und Name des Entpackprogramms, mit
[M26]   24: ;dem jedes gepackte Programm beginnt
[KZ1]   25: dnum1        dw Ø
[MZ1]   26: dnum2        dw Ø
[PZ1]   27: dnum3        dw Ø
[YY6]   28: ;Speicher für Dateinummern von file1-file3
[293]   29: buff1        db 7ØØØ dup(Ø)
[493]   3Ø: buff2        db 7ØØØ dup(Ø)
[ES6]   31: ;Pufferspeicher für Lese- und Schreib-
[YT4]   32: ;zugriffe auf Speichermedien
[VØ3]   33: spec         dw 1ØØ dup(Ø)
[KX5]   34: ;Zählt Anzahl jedes Bytes in file1
[C23]   35: specx        db 1ØØ dup(Ø)
[WW4]   36: ;Speicher der Decodiertabelle
[ØØØ]   37:
[7Z1]   38: long3        dw Ø
[T54]   39: ;Sichert Länge von file3
[5V1]   4Ø: crc          dw Ø
[Y87]   41: ;Sichert Checksumme des Orginalprogrammes
[2N2]   42: packlaenge   dw 1ØØ
[AP6]   43: ;Sichert Länge des gepackten Programms
[T82]   44: longx        dw Ø,Ø
[2W5]   45: ;Sichert Zeigerposition auf Datei
[422]   46: header       dw Ø
[ME3]   47: ;Sichert Kopf-länge
[ØV2]   48: bitcount     dw Ø,Ø,Ø
[GY6]   49: ;Sichert Bit/Bytepositionen beim Packen
[CW5]   5Ø: msg1         db Øa,'Dieses Programm macht '
[L15]   51:              db 'aus COM und EXE Programme'
[R44]   52:              db 'n',Øa,Ødh,'ein gepacktes '
[D44]   53:              db 'COM Programm.',Øa,Ødh,Øa
[DH4]   54:              db 'Syntax: C Datei1 Datei2  '
[7H4]   55:              db '-->  Datei1=Input  Datei2='
[D23]   56:              db 'Output',Øa,Ødh,24
[G85]   57: msg2         db 'Datei Fehler!',Øa,Ødh,24
[BU5]   58: msg3         db 'Exe-Datei zu lang!',Øa,Ødh
[8LØ]   59:              db 24
[756]   6Ø: msg4         db 'Es dürfen keine Exe-file '
[8X4]   61:              db 'Relokationen vorhanden se'
[AF4]   62:              db 'in!',Øa,Ødh,'Programm mit'
[6E4]   63:              db ' /e Option neu linken.',Øa
[851]   64:              db Ødh,24
[4D2]   65: dataseg      ends
[ØØØ]   66:
[TW2]   67: modul1       segment
[RV6]   68:              assume cs:modul1,ds:dataseg,ss:
            stackseg
[ZY3]   69: start:       mov ax,dataseg
[3H1]   7Ø:              mov es,ax
[ULØ]   71:              cld
[8E1]   72:              mov si,8Ø
[135]   73: ;DS:SI auf Eingabezeile setzen
[CVØ]   74:              lodsb
[ØF1]   75:              xor ah,ah
[161]   76:              or al,al
[CB1]   77:              jnz st1
[DD3]   78: syntaxerr:   push es
[E31]   79:              pop ds
[BD1]   8Ø:              mov ah,Ø9
[RU1]   81:              lea dx,msg1
[3ZØ]   82:              int 21
[7U1]   83:              mov ax,4cØ1
[3ZØ]   84:              int 21
[GL6]   85: ;Eingabezeile enthält nichts oder eine
[ØU4]   86: ;syntaktisch falsche Eingabe
[GZ2]   87: ;(Return-Code = Ø1)
[ØØØ]   88:
[EU2]   89: st1:         mov cx,ax
[ZW1]   9Ø:              lea.di,file1
[G42]   91: st2:         lodsb
[NB1]   92:              cmp al,2Ø
[LXØ]   93:              jz st2
[4Ø1]   94:              dec si
[M42]   95: st3:         lodsb
[NB1]   96:              cmp al,2Ø
[WB1]   97:              jnz st5
[P53]   98:              cmp di,offset file2
[RVØ]   99:              ja st7
[ØX1]  1ØØ:              lea di,file2
[V72]  1Ø1:              jmp short st6
[3X2]  1Ø2: st5:         cmp al,Ødh
[ZXØ]  1Ø3:              jz st6
[TB1]  1Ø4:              cmp al,6Ø
[HVØ]  1Ø5:              jb st4
[WH1]  1Ø6:              and al,Ødf
[3N2]  1Ø7: st4:         cmp al,21
[L32]  1Ø8:              jb syntaxerr
[YC1]  1Ø9:              cmp al,5c
[G32]  11Ø:              ja syntaxerr
[NE3]  111:              cmp di,offset file2+2Ø
[V52]  112:              jz syntaxerr
[NXØ]  113:              stosb
[1S2]  114: st6:         loop st3
[EF4]  115: st7:         cmp di,offset file2
[ZA2]  116:              jbe syntaxerr
[WF6]  117: ;Beide Programmnamen werden eingelesen
[146]  118: ;und auf ihre Syntax hin überprüft
[LT6]  119: ;(maximale Programmnamenlänge = 32 Bytes)
[ØØØ]  12Ø:
[EC1]  121:              push es
[E31]  122:              pop ds
[SE2]  123:              call openfiles
[UE2]  124:              call writefile
[PP2]  125:              call closefiles
[3U1]  126:              mov ax,4cØØ
[3ZØ]  127:              int 21
[AX4]  128: ;Programm erfolgreich beendet
[DZ2]  129: ;(Return-Code = ØØ)
[ØØØ]  13Ø:
[AD3]  131: openfiles    proc near
[3Ø4]  132: ;Öffnet file1 bis file3
[4U1]  133:              mov ax,3dØØ
[CY1]  134:              lea dx,file1
[3ZØ]  135:              int 21
[7T2]  136:              mov ds:dnum1,ax
[232]  137:              jnc openfil1
[LD5]  138: ;Orginalprogramm wird geöffnet
[ØØØ]  139:
[883]  14Ø: err1:        lea dx,msg2
[MX2]  141: err1a:       mov ah,Ø9
[3ZØ]  142:              int 21
[BU1]  143:              mov ax,4cØ2
[3ZØ]  144:              int 21
[857]  145: ;Ein Speichermedien-Fehler ist aufgetreten
[LZ2]  146: ;(Return-Code = Ø2)
[ØØØ]  147:
[E73]  148: openfil1:    mov ah,3c
[DY1]  149:              lea dx,file2
[FE1]  15Ø:              mov cx,2Ø
[3ZØ]  151:              int 21
[8T2]  152:              mov ds:dnum2,ax
[Ø41]  153:              jc err1
[9Y6]  154: ;Zieldatei wird kreiert (sollte bereits
[2M6]  155: ;ein Programm diesen Namen haben, so wird
[DA6]  156: ;es ohne Fehlermeldung überschrieben)
[4U1]  157:              mov ax,3dØØ
[EY1]  158:              lea dx,file3
[3ZØ]  159:              int 21
[9T2]  16Ø:              mov ds:dnum3,ax
[Ø41]  161:              jc err1
[3PØ]  162:              ret
[6K2]  163: openfiles    endp
[ØØØ]  164:
[AD3]  165: writefile    proc near
[TR4]  166: ;Zieldatei wird beschrieben
[8E1]  167:              mov ah,3f
[6M2]  168:              mov bx,ds:dnum1
[YK1]  169:              mov cx,1ØØ
[LY1]  17Ø:              lea dx,buff2
[3ZØ]  171:              int 21
[441]  172:              jc err2
[VL5]  173: ;Orginalprogramm wird ausgelesen
[1T1]  174:              mov ax,42ØØ
[PH1]  175:              xor cx,cx
[TH1]  176:              xor dx,dx
[3ZØ]  177:              int 21
[441]  178:              jc err2
[556]  179: ;und der Dateizeiger zurückgesetzt
[8E1]  18Ø:              mov ah,3f
[EM2]  181:              mov bx,ds:dnum3
[4T1]  182:              mov cx,7ØØØ
[KY1]  183:              lea dx,buff1
[3ZØ]  184:              int 21
[441]  185:              jc err2
[N84]  186: ;PACK2.COM wird eingelesen
[6S2]  187:              mov ds:long3,ax
[M42]  188:              lea di,buff1+3
[412]  189:              mov ax,Øfffe
[G63]  19Ø:              stosw                    ;SP
[GH1]  191:              xor ax,ax
[T63]  192:              stosw                    ;SS
[UK1]  193:              mov ax,1ØØ
[W53]  194:              stosw                    ;IP
```

Listing 3. Der Assembler-Quelltext zu »pack.exe«

```
[GH1]  195:                xor ax,ax
[T53]  196:                stosw                          ;CS
[W76]  197: ;Register werden auf Standart-Format
[1X2]  198: ;( = *.COM) gesetzt
[4Y1]  199:                lea si,buff2
[PWØ]  2ØØ:                lodsw
[KX1]  2Ø1:                cmp ax,5a4dh
[ZT1]  2Ø2:                jnz write1
[FV6]  2Ø3: ;Ist das Orginalprogramm im *.EXE-Format?
[PWØ]  2Ø4:                lodsw
[PWØ]  2Ø5:                lodsw
[5D1]  2Ø6:                cmp ax,83
[9D1]  2Ø7:                jb write2
[ØØØ]  2Ø8:
[E83]  2Ø9: err3:          lea dx,msg3
[PK1]  21Ø:                jmp errla
[983]  211: err2:          lea dx,msg2
[PK1]  212:                jmp errla
[L83]  213: err4:          lea dx,msg4
[PK1]  214:                jmp errla
[UG2]  215: write2:        lodsw
[571]  216:                or ax,ax
[2G1]  217:                jnz err4
[PWØ]  218:                lodsw
[PP2]  219:                mov ds:header,ax
[PWØ]  22Ø:                lodsw
[DX1]  221:                cmp ax,Øfff
[241]  222:                ja err3
[PWØ]  223:                lodsw
[PWØ]  224:                lodsw
[9A1]  225:                add ax,1Ø
[PC1]  226:                push ax
[M42]  227:                lea di,buff1+3
[H63]  228:                movsw                          ;SP
[R31]  229:                pop ax
[T63]  23Ø:                stosw                          ;SS
[PWØ]  231:                lodsw
[X53]  232:                movsw                          ;IP
[PWØ]  233:                lodsw
[9A1]  234:                add ax,1Ø
[T53]  235:                stosw                          ;CS
[396]  236: ;*.EXE-Format wird aus Orginalprogramm
[1D4]  237: ;ausgelesen und gesichert
[F23]  238: write3:        mov ah,3f
[6M2]  239:                mov bx,ds:dnum1
[YK1]  24Ø:                mov cx,1ØØ
[LY1]  241:                lea dx,buff2
[3ZØ]  242:                int 21
[441]  243:                jc err2
[AR2]  244:                mov ax,ds:header
[SC1]  245:                sub ax,1Ø
[PP2]  246:                mov ds:header,ax
[571]  247:                or ax,ax
[5U1]  248:                jnz write3
[MZ5]  249: ;*.EXE-Programmkopf wird überlesen
[YØ3]  25Ø: write1:        mov ah,4Ø
[KY1]  251:                lea dx,buff1
[AM2]  252:                mov bx,ds:dnum2
[HM2]  253:                mov cx,ds:long3
[3ZØ]  254:                int 21
[441]  255:                jc err2
[LB2]  256:                mov ds:long3,Ø
[5T1]  257:                mov ax,42Ø1
[PH1]  258:                xor cx,cx
[TH1]  259:                xor dx,dx
[6M2]  26Ø:                mov bx,ds:dnum1
[3ZØ]  261:                int 21
[G41]  262:                jc err5
[BT2]  263:                mov ds:longx,ax
[ØU2]  264:                mov ds:longx+2,dx
[K23]  265: write4:        mov ah,3f
[6M2]  266:                mov bx,ds:dnum1
[4T1]  267:                mov cx,7ØØØ
[KY1]  268:                lea dx,buff1
[3ZØ]  269:                int 21
[FR1]  27Ø:                jnc write5
[NL5]  271: ;Orginalprogramm wird eingelesen
[6W2]  272: err5:          jmp err2
[K84]  273: write5:        add ds:long3,ax
[RR1]  274:                call count
[767]  275: ;und die Häufigkeit jedes Bytes gezählt
[ER1]  276:                cmp ax,7ØØØ
[NF1]  277:                jz write4
[1T1]  278:                mov ax,42ØØ
[7T2]  279:                mov dx,ds:longx
[MX2]  28Ø:                mov cx,ds:longx+2
[6M2]  281:                mov bx,ds:dnum1
[3ZØ]  282:                int 21
[G41]  283:                jc err5
[V97]  284: ;Anschließend Dateizeiger zurücksetzen
[ØØØ]  285:
[EN1]  286:                call table
[ZD7]  287: ;Bytehäufigkeit in Decodiertabelle wandeln
[BG1]  288:                xor bp,bp
[DY1]  289:                lea ax,buff2
[6F3]  29Ø:                mov ds:bitcount+2,ax
[BY1]  291:                call savtab
[RØ7]  292: ;Decodiertabelle auf Zieldatei übertragen
[S23]  293: write6:        mov ah,3f
[6M2]  294:                mov bx,ds:dnum1
[4T1]  295:                mov cx,7ØØØ
[KY1]  296:                lea dx,buff1
[3ZØ]  297:                int 21
[G41]  298:                jc err5
[T62]  299:                call changsav
[ER1]  3ØØ:                cmp ax,7ØØØ
[TF1]  3Ø1:                jz write6
[FE2]  3Ø2:                call changrest
[277]  3Ø3: ;Orginalprogramm wird Schritt für Schritt
[B27]  3Ø4: ;eingelesen, gepackt und auf die Zieldatei
[FA2]  3Ø5: ;übertragen
[1T1]  3Ø6:                mov ax,42ØØ
[PH1]  3Ø7:                xor cx,cx
[BP1]  3Ø8:                mov dx,Øbh
[AM2]  3Ø9:                mov bx,ds:dnum2
[3ZØ]  31Ø:                int 21
[G41]  311:                jc err5
[EZ1]  312:                lea dx,long3
[XC1]  313:                mov ah,4Ø
[TE1]  314:                mov cx,Ø6
[3ZØ]  315:                int 21
[G41]  316:                jc err5
[EY6]  317: ;Arbeitsvariablen von PACK2.COM auf Ziel-
[CV2]  318: ;datei übertragen
[3PØ]  319:                ret
[6K2]  32Ø: writefile      endp
[ØØØ]  321:
[N13]  322: count          proc near
[1D6]  323: ;Zählt die Häufigkeit jedes Bytes in
[U82]  324: ;einer Tabelle
[PC1]  325:                push ax
[TC1]  326:                push bx
[WC1]  327:                push cx
[2C1]  328:                push dx
[GC1]  329:                push si
[3B1]  33Ø:                push di
[EH1]  331:                mov cx,ax
[3Y1]  332:                lea si,buff1
[4G2]  333: count1:        lodsb
[ØF1]  334:                xor ah,ah
[UC1]  335:                shl ax,Ø1
[CH1]  336:                mov bx,ax
[FZ2]  337:                add bx,offset spec
[CL3]  338:                inc word ptr ds:[bx]
[7Ø2]  339:                loop count1
[G21]  34Ø:                pop di
[E31]  341:                pop si
[X31]  342:                pop dx
[V31]  343:                pop cx
[T31]  344:                pop bx
[R31]  345:                pop ax
[3PØ]  346:                ret
[H62]  347: count          endp
[ØØØ]  348:
[1Y2]  349: table          proc near
[ZD7]  35Ø: ;Bytehäufigkeit in Decodiertabelle wandeln
[HE1]  351:                mov cx,Ø2
[G73]  352:                mov bp,11111ØØØØØØØØØØØb
[4Y1]  353:                lea di,specx
[XX2]  354: tableØ:        push cx
[7W2]  355: table1:        push di
[722]  356:                call highest
[VG1]  357:                mov ax,bp
[ZYØ]  358:                stosw
[FG1]  359:                mov ax,di
[G21]  36Ø:                pop di
[U83]  361:                sub ax,offset spec+2
[KD1]  362:                shr ax,Ø1
[NXØ]  363:                stosb
[VG1]  364:                mov ax,bp
[WØ1]  365:                inc bp
[YX1]  366:                loop table1
[ØØØ]  367:
[MF1]  368:                xor al,al
[2Ø1]  369:                inc ah
[MG1]  37Ø:                mov bp,ax
[V31]  371:                pop cx
[872]  372:                cmp ah,11111ØØ1b
[LC1]  373:                jz tableØ
[UB1]  374:                shl cl,Ø1
[3M1]  375:                jnc tableØ
[3PØ]  376:                ret
[ØØØ]  377:
[S53]  378: highest:       push cx
[CF6]  379: ;Sucht aus Bytehäufigkeitstabelle das
[Y84]  38Ø: ;nächsthöchste Element
[LH1]  381:                xor bx,bx
[AW1]  382:                lea si,spec
[YK1]  383:                mov cx,1ØØ
[XP2]  384: highest1:      lodsw
[PC1]  385:                push ax
[L52]  386:                and ah,11111ØØØb
[472]  387:                cmp ah,11111ØØØb
[R31]  388:                pop ax
[ØX1]  389:                jz highest2
[TF1]  39Ø:                cmp ax,bx
[SU1]  391:                jb highest2
[CH1]  392:                mov bx,ax
[RF1]  393:                mov di,si
```

```
[X14]  394: highest2:   loop highest1
[ØØØ]  395:
[6ZØ]  396:             dec di
[6ZØ]  397:             dec di
[V31]  398:             pop cx
[3PØ]  399:             ret
[X22]  4ØØ: table       endp
[ØØØ]  4Ø1:
[N33]  4Ø2: savtab      proc near
[A55]  4Ø3: ;Decodiertabelle auf Zieldatei
[BR3]  4Ø4: ;übertragen (Buffer)
[WC1]  4Ø5:             push cx
[NE1]  4Ø6:             mov cx,8Ø
[2Z1]  4Ø7:             lea si,specx
[LB3]  4Ø8:             mov di,ds:bitcount+2
[LN1]  4Ø9:             rep movsw
[3F3]  41Ø:             mov ds:bitcount+2,di
[V31]  411:             pop cx
[3PØ]  412:             ret
[H82]  413: savtab      endp
[ØØØ]  414:
[983]  415: changsav    proc near
[K96]  416: ;Orginalprogrammteile packen und auf
[VL3]  417: ;Zieldatei übertragen
[PC1]  418:             push ax
[TC1]  419:             push bx
[WC1]  42Ø:             push cx
[ZC1]  421:             push dx
[TB1]  422:             push bp
[EH1]  423:             mov cx,ax
[3Y1]  424:             lea si,buff1
[653]  425:             mov bp,ds:bitcount
[LB3]  426:             mov di,ds:bitcount+2
[WC3]  427:             mov ax,ds:bitcount+4
[NT4]  428: ;Bit/Byte-Position einlesen
[HB3]  429: changsav1:  mov dx,ax
[CVØ]  43Ø:             lodsb
[WC1]  431:             push cx
[GC1]  432:             push si
[ØF1]  433:             xor ah,ah
[S12]  434:             add ds:crc,ax
[C76]  435: ;Für Checksumme jedes Byte addieren
[UC1]  436:             shl ax,Ø1
[DZ2]  437:             add ax,offset spec
[1H1]  438:             mov si,ax
[PWØ]  439:             lodsw
[BR5]  44Ø: ;Wert aus Decodiertabelle einlesen
[ZR1]  441:             xchg ax,dx
[ZC1]  442:             push dx
[8C1]  443:             shr dh,Ø1
[ZXØ]  444:             pushf
[8C1]  445:             shr dh,Ø1
[ZXØ]  446:             pushf
[8C1]  447:             shr dh,Ø1
[ZXØ]  448:             pushf
[3U6]  449: ;die drei höchsten Bits Kennzeichnen die
[GD5]  45Ø: ;Anzahl der noch kommenden Bits
[FD1]  451:             mov cl,Ø3
[XP1]  452:             call rclax
[C27]  453: ;CL Bits aus dem Stack holen und speichern
[X31]  454:             pop dx
[R52]  455:             and dh,ØØØØØ111b
[461]  456:             or dh,dh
[V72]  457:             jnz changsav2
[8Ø1]  458:             inc dh
[G93]  459: changsav2:  mov cl,dh
[6F1]  46Ø:             xor ch,ch
[3H1]  461:             mov si,cx
[T63]  462: changsav3:  shr dl,Ø1
[ZXØ]  463:             pushf
[YD2]  464:             loop changsav3
[2H1]  465:             mov cx,si
[XP1]  466:             call rclax
[C27]  467: ;CL Bits aus dem Stack holen und speichern
[E31]  468:             pop si
[V31]  469:             pop cx
[SD2]  47Ø:             loop changsav1
[ØØØ]  471:
[R83]  472:             mov ds:bitcount,bp
[3F3]  473:             mov ds:bitcount+2,di
[AF3]  474:             mov ds:bitcount+4,ax
[8C5]  475: ;neue Bit/Byte-Position sichern
[131]  476:             pop bp
[X31]  477:             pop dx
[V31]  478:             pop cx
[T31]  479:             pop bx
[R31]  48Ø:             pop ax
[3PØ]  481:             ret
[ØØØ]  482:
[TM2]  483: rclax:      pop dx
[2C2]  484: rclax3:     popf
[5B1]  485:             rcl al,Ø1
[WØ1]  486:             inc bp
[HC1]  487:             cmp bp,Ø8
[MA1]  488:             jb rclax1
[BG1]  489:             xor bp,bp
[NXØ]  49Ø:             stosb
[YV2]  491:             inc ds:packlaenge
[MU5]  492: ;Zählt Anzahl der gepackten Bytes
[MF1]  493:             xor al,al
[2P3]  494:             cmp di,offset buff2+7ØØØ
[7T1]  495:             jnz rclax1
[L15]  496: ;Ist der Buffer bereits voll?
[PC1]  497:             push ax
[TC1]  498:             push bx
[WC1]  499:             push cx
[ZC1]  5ØØ:             push dx
[XC1]  5Ø1:             mov ah,4Ø
[AM2]  5Ø2:             mov bx,ds:dnum2
[4T1]  5Ø3:             mov cx,7ØØØ
[LY1]  5Ø4:             lea dx,buff2
[6G1]  5Ø5:             mov di,dx
[3ZØ]  5Ø6:             int 21
[EP1]  5Ø7:             jnc rclax2
[7W2]  5Ø8: err6:       jmp err2
[9R2]  5Ø9: rclax2:     pop dx
[V31]  51Ø:             pop cx
[T31]  511:             pop bx
[R31]  512:             pop ax
[XV5]  513: ;Buffer auf Speichermedium sichern
[PK2]  514: rclax1:     dec cl
[DT1]  515:             jnz rclax3
[ZC1]  516:             push dx
[3PØ]  517:             ret
[ØØØ]  518:
[HR1]  519: changrest:
[AG5]  52Ø: ;restlichen Teil des Buffers auf
[984]  521: ;Speichermedium sichern
[PC1]  522:             push ax
[TC1]  523:             push bx
[WC1]  524:             push cx
[ZC1]  525:             push dx
[3B1]  526:             push di
[WC3]  527:             mov ax,ds:bitcount+4
[Ø63]  528:             mov cx,ds:bitcount
[LB3]  529:             mov di,ds:bitcount+2
[7C1]  53Ø:             sub cl,Ø8
[HØ1]  531:             neg cl
[3F1]  532:             shl ax,cl
[NXØ]  533:             stosb
[YV2]  534:             inc ds:packlaenge
[XC1]  535:             mov ah,4Ø
[AM2]  536:             mov bx,ds:dnum2
[LG1]  537:             mov cx,di
[3B3]  538:             sub cx,offset buff2-1
[LY1]  539:             lea dx,buff2
[3ZØ]  54Ø:             int 21
[M41]  541:             jc err6
[G21]  542:             pop di
[X31]  543:             pop dx
[V31]  544:             pop cx
[T31]  545:             pop bx
[R31]  546:             pop ax
[3PØ]  547:             ret
[5D2]  548: changsav    endp
[ØØØ]  549:
[ØH3]  55Ø: closefiles  proc near
[WV5]  551: ;Alle Programme wieder schließen
[6E1]  552:             mov ah,3e
[6M2]  553:             mov bx,ds:dnum1
[3ZØ]  554:             int 21
[6E1]  555:             mov ah,3e
[AM2]  556:             mov bx,ds:dnum2
[3ZØ]  557:             int 21
[6E1]  558:             mov ah,3e
[EM2]  559:             mov bx,ds:dnum3
[3ZØ]  56Ø:             int 21
[3PØ]  561:             ret
[WN2]  562: closefiles  endp
[ØØØ]  563:
[E62]  564: modul1      ends
[4K1]  565:             end start
```

Listing 3. Der Assembler-Quelltext zu »pack.exe«

```
[T6A]    1: title Entpackprogramm, mit dem jedes gepack
            te Programm beginnt
[VL1]    2: name pack2
[ØØØ]    3:
[YV3]    4: ;Autor: Richard Lesser
[3P4]    5: ;Copyright Everts & Hagedorn
[3E4]    6: ;DOS International Extra 7
[ØØØ]    7:
[X64]    8: ;Assemblieren und Linken:
[TW1]    9: ;masm pack2;
[DX1]   1Ø: ;link pack2;
[CE4]   11: ;exe2bin pack2.exe pack2.com
[162]   12: ;del pack2.exe
[ØØØ]   13:
[XG1]   14:             .radix 1Øh
[WW2]   15: modul2      segment
[8S2]   16:             assume cs:modul2
[241]   17:             org 1ØØ
[ØØØ]   18:
[SY2]   19: start:      jmp main
```

Listing 4. »pack2.com« als Assembler-Quelltext

```
[ØØØ]   2Ø:
[713]   21: daten1      proc near
[182]   22: relss       dw Ø,Ø
[9X4]   23: ;Relocationsoffset für SS:SP
[H72]   24: relcs       dw Ø,Ø
[EP2]   25: ;und für CS:IP
[R12]   26: laenge      dw Ø
[KB6]   27: ;Länge des Orginalprogramms in Bytes
[5V1]   28: crc         dw Ø
[BR5]   29: ;Checksumme des Orginalprogramms
[HD2]   3Ø: packlaenge  dw Ø
[F95]   31: ;Länge des gepackten Programms
[H82]   32:             db 'R.Lesser / '
[KX3]   33:             db 'DOS International',Ø
[362]   34: daten1      endp
[ØØØ]   35:
[RØ3]   36: haupt       proc near
[BC1]   37:             push ds
[EC1]   38:             push es
[E31]   39:             pop ds
[ZK2]   4Ø:             lea si,mainende
[PD1]   41:             sub di,di
[NE1]   42:             mov cx,8Ø
[LN1]   43:             rep movsw
[ES4]   44: ;Decodiertabelle wird auf den
[ØA4]   45: ;Segmentanfang verschoben
[G31]   46:             pop es
[131]   47:             pop bp
[LH1]   48:             mov di,1ØØ
[EMØ]   49:             cli
[3K3]   5Ø:             mov ds:hauptadrw,sp
[PWØ]   51:             lodsw
[NM1]   52:             xchg ah,al
[VD1]   53:             mov dl,Ø8
[DV2]   54:             jmp short haupt3
[M46]   55: ;Variablen werden initialisiert und
[GZ4]   56: ;Sprung in die Entpackroutine
[R33]   57: haupt5a:    mov dl,Ø8
[CVØ]   58:             lodsb
[FV2]   59:             jmp short haupt5
[U33]   6Ø: haupt6a:    mov dl,Ø8
[CVØ]   61:             lodsb
[GV2]   62:             jmp short haupt6
[X33]   63: haupt7a:    mov dl,Ø8
[CVØ]   64:             lodsb
[HV2]   65:             jmp short haupt7
[343]   66: haupt9a:    mov dl,Ø8
[CVØ]   67:             lodsb
[LV2]   68:             jmp short haupt9
[N33]   69: haupt4a:    sub al,al
[KØ1]   7Ø:             inc cl
[EV2]   71:             jmp short haupt4
[ØØØ]   72:
[223]   73: haupt3:     sub cl,cl
[UC1]   74:             shl ax,Ø1
[9B1]   75:             rcl cl,Ø1
[FZØ]   76:             dec dl
[CN1]   77:             jz haupt5a
[Ø13]   78: haupt5:     shl ax,Ø1
[9B1]   79:             rcl cl,Ø1
[FZØ]   8Ø:             dec dl
[EN1]   81:             jz haupt6a
[313]   82: haupt6:     shl ax,Ø1
[9B1]   83:             rcl cl,Ø1
[FZØ]   84:             dec dl
[GN1]   85:             jz haupt7a
[416]   86: ;ersten drei Bits (Kennbits) werden
[FS4]   87: ;aus AX herausgerollt in CL
[263]   88: haupt7:     mov sp,ax
[BE1]   89:             mov ax,Ø1
[B61]   9Ø:             or cl,cl
[AN1]   91:             jz haupt4a
[633]   92: haupt4:     shl ax,cl
[CH1]   93:             mov bx,ax
[CH1]   94:             mov ax,sp
[8F1]   95:             mov ch,cl
[ØØØ]   96:
[2E1]   97:             sub cl,cl
[913]   98: haupt8:     shl ax,Ø1
[9B1]   99:             rcl cl,Ø1
[FZØ]  1ØØ:             dec dl
[MN1]  1Ø1:             jz haupt9a
[LL2]  1Ø2: haupt9:     dec ch
[DU1]  1Ø3:             jnz haupt8
[HU5]  1Ø4: ;diese Routine rotiert aus AX genau
[UH2]  1Ø5: ;CH Bits nach CL
[DD1]  1Ø6:             add bx,cx
[6H2]  1Ø7:             mov cl,ds:[bx]
[A86]  1Ø8: ;Byte wird aus Decodiertabelle gelesen
[942]  1Ø9:             sub ds:crc,cx
[EØ7]  11Ø: ;Checksumme wird aus jedem Byte berechnet
[KH2]  111:             mov es:[di],cl
[524]  112: ;Byte wird abgespeichert
[BØ1]  113:             inc di
[RZØ]  114:             dec bp
[YT1]  115:             jnz haupt3
[7K6]  116: ;BP = Anzahl noch zu entpackender Bytes
[ØØØ]  117:
[3K3]  118:             mov sp,ds:hauptadrw
```

```
[6PØ]  119:             sti
[N52]  12Ø:             mov ax,ds:crc
[571]  121:             or ax,ax
[HL1]  122:             jz haupt1Ø
[8X4]  123: ;Ist die Checksumme korrekt?
[VU1]  124:             lea dx,msg2
[ZU2]  125:             jmp short error1
[RC5]  126: ;Nein, Fehler 'CHECKSUMMEN-FEHLER'
[ØØØ]  127:
[UZ2]  128: haupt1Ø:    push es
[E31]  129:             pop ds
[R31]  13Ø:             pop ax
[5H1]  131:             mov bx,ds
[AD1]  132:             add ax,bx
[XH3]  133:             mov cs:hauptadrw+2,ax
[R31]  134:             pop ax
[BH3]  135:             mov cs:hauptadrw,ax
[4V6]  136: ;Sprungadresse auf Programmstart wird in
[ØL6]  137: ;Variable gespeichert (mit Far-Attribut)
[T31]  138:             pop bx
[V31]  139:             pop cx
[R31]  14Ø:             pop ax
[9H1]  141:             mov dx,ds
[ED1]  142:             add bx,dx
[EMØ]  143:             cli
[SH1]  144:             mov sp,cx
[ØK1]  145:             mov ss,bx
[6PØ]  146:             sti
[YN4]  147: ;Stack (SP:SS) des entpackten
[874]  148: ;Programms wird gesetzt
[NT2]  149:             jmp cs:hauptadr
[L52]  15Ø: haupt       endp
[ØØØ]  151:
[A13]  152: daten2      proc near
[FA4]  153: msg1        db 'OUT OF MEMORY!'
[N54]  154: hauptadrw   equ this word
[A93]  155: hauptadr    dd 24Ø7Ød9a
[LØ6]  156: msg2        db 'CHECKSUMMEN-FEHLER!',Øa,24
[6U2]  157: ;Fehlermeldungen
[662]  158: daten2      endp
[ØØØ]  159:
[7W2]  16Ø: main        proc near
[ULØ]  161:             cld
[ØF3]  162:             mov si,offset relss
[TE1]  163:             mov cx,Ø6
[RU2]  164: main2:      push ax
[PWØ]  165:             lodsw
[RN1]  166:             loop main2
[NV3]  167: ;Daten auf Stack sichern
[ZD1]  168:             mov ah,4a
[WS1]  169:             mov bx,1ØØØ
[3ZØ]  17Ø:             int 21
[AA1]  171:             jc error
[WR6]  172: ;Größe des Programmsegment verändern
[K75]  173: ;BX=Neue Größe in Paragraphen
[3H3]  174: ;1 Paragraph = 16 Bytes
[DD1]  175:             mov ah,48
[2T1]  176:             mov bx,1Ø2Ø
[3ZØ]  177:             int 21
[VE1]  178:             jnc main1
[LU6]  179: ;Neues Segment anfordern (Arbeitssegment)
[T27]  18Ø: ;BX=Gewünschte Größe in Paragraphen
[ØØØ]  181:
[ØD3]  182: error:      lea dx,msg1
[123]  183: error1:     mov ah,Ø9
[3ZØ]  184:             int 21
[7U1]  185:             mov ax,4cØ1
[3ZØ]  186:             int 21
[LK6]  187: ;Ein Fehler ist aufgetreten: Abbrechen
[M63]  188: ;mit Return-Code Ø1
[ØØØ]  189:
[YZ4]  19Ø: main1:      mov ds:hauptadrw+2,ax
[T65]  191:             mov ds:hauptadrw,offset haupt
[3H1]  192:             mov es,ax
[FD1]  193:             mov ah,49
[3ZØ]  194:             int 21
[AA1]  195:             jc error
[HA4]  196: ;Segment wieder freigeben
[ULØ]  197:             cld
[HK1]  198:             mov si,1ØØ
[RF1]  199:             mov di,si
[284]  2ØØ:             mov cx,offset mainende-1ØØ
[2C3]  2Ø1:             add cx,ds:packlaenge
[P11]  2Ø2:             inc cx
[PD1]  2Ø3:             shr cx,Ø1
[LN1]  2Ø4:             rep movsw
[3M5]  2Ø5: ;Entpackprogramm wird verschoben
[Z42]  2Ø6:             jmp hauptadr
[8Y4]  2Ø7: ;Far-Sprung ins neue Segment
[ØØØ]  2Ø8:
[VØ4]  2Ø9: mainende    equ this word
[312]  21Ø: main        endp
[ØØØ]  211:
[H62]  212: modul2      ends
[4K1]  213:             end start
```

Listing 4. »pack2.com« als Assembler-Quelltext

Kurven plotten ganz einfach

Manchmal ist man es leid: Die Augen flimmern, und der Kopf brummt vor lauter buntfarbigen Feldern, Pulldown-Menüs, Maus-Flächen, Fenstern und Befehlsleisten. Man möchte statt High-Tech oft auch einmal eine einfache Hilfe bei der grafischen Darstellung von Zahlen.

Es geht um ein altbekanntes Thema: Gegeben ist eine Reihe von Zahlen, gewünscht ein einfaches Diagramm. Sicher, dafür gibt es mächtige komfortable Riesen, Lotus 1-2-3, Asyst, Chart, HPG, Plotit und viele mehr. Sie bieten Torten, Balken, dreidimensionale Ansichten und einiges mehr – aber wie plottet man mal eben 2000 Zahlen? Ganz abgesehen von den ebenfalls dreidimensionalen Kosten solcher Pakete führt der überflüssige Luxus oft zum Überdruß.

Bei Physikern, Biologen, Mathematikern – überhaupt in Naturwissenschaft, Medizin und Technik fallen ständig Datenpakete an, die man rasch ansehen und auch mit einfachen Matrixdruckern festhalten möchte. Oft fehlt eine simple Möglichkeit, rasch einen Zahlensatz als Kurve anzusehen und zu zeichnen. »Kurplot« ist nun ein wirklich einfaches Programm (fast altmodisch, sogar in Basic), einfacher geht's kaum. Endloses Einstellen von Parametern und das Studium von dicken Handbüchern ist damit überflüssig.

Was kann das Programm? Jeder einfache ASCII-Datensatz (hier maximal 6000 Fest- oder Gleitkommazahlen, jede für sich in einer Zeile) wird als Kurve, Histogramm oder Punktreihe dargestellt. Der Schirm wird mit 640x200 Bildpunkten voll ausgenützt und kann auf einem Drucker mit der Taste [PrtScr] sofort ausgedruckt werden. Alle vier Seiten des Rahmens zeigen Skalenstriche. Benötigt werden MS-DOS ab Version 2.0 und eine Grafikkarte (hier CGA); mit anderen Karten kann die Auflösung verbessert werden.

Nun zur Bedienung von Kurplot. Eine vorhandene Datei mit den darzustellenden Werten wird mit bis zu 640 Punkten direkt abgebildet. Sind es mehr als 640 Daten, kann man entweder mehrere Teile plotten oder durch Mittelung die Punktmenge reduzieren.

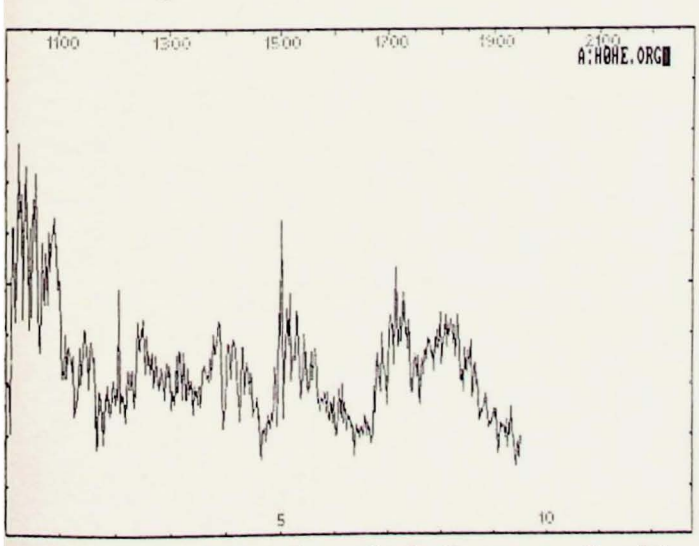


Bild 1. Eine typische Druckausgabe des Programms »Kurplot«

Für das Ganze sind (höchstens) sieben Eingaben erforderlich: zum ersten die Wahl zwischen Kurve, Histogramm oder Punkten und dann der Name der Datei; Pfadangaben sind erlaubt.

Nun folgen einige Angaben zur Form des Plots: Mittelwertbildung, wenn mehr als 640 Daten auf nur einer Seite abzubilden sind. Der ganzzahlige Faktor M (>1) reduziert die Datenzahl auf 1/M. Es wird nur jeder M-te Wert, (Stufenmittel mit den folgenden Originalnachbarn) gezeichnet. Jetzt wählen Sie die Datendarstellung: nur positive Zahlen (linear oder y-logarithmisch) oder Zahlen beiderlei Vorzeichens (Mittelhorizontale als Nullinie).

Als Beispiel: Bild 1 zeigt das Ergebnis der Eingaben aus Bild 2. Es folgt die Angabe des x- und y-Skalenfaktors (»automatisch« gibt bestmögliche Dehnung). Der x-Faktor muß ganzzahlig sein, sonst wird er gerundet. Für die richtige Wahl zeigt der Bildschirm den Maximal- und den Minimalwert der Datei und die Bildschirmgrenzen an. Zu große Werte bleiben außerhalb des Bildschirms unsichtbar. Mit der Wahl eines Datenausschnittes bei mehr als 640 Daten starker horizontaler Dehnung wird der gezeigte Datenauschnitt definiert. Dazu ist die Eingabe des Anfangswertes K1 und des Endwertes K2 erforderlich, die durch ein Komma getrennt werden müssen.

Nun zur Programmbeschreibung: Anhand des Programmtextes »kurplot.bas« aus Listing 1 lassen sich Eingaben und Ablauf verfolgen. Nach der Wahl der Kurvenform wird der Dateiname abgefragt, und die Daten werden gelesen. Wenn die Daten gespeichert sind, erscheint zur Kontrolle auf dem Bildschirm die Datenzahl, und ein Piepston wird ausgegeben. Je nach Rechnergeschwindigkeit dauert dies bei 1000 Daten zwischen 3 und 30 Sekunden. Nun wird nach einer Mittelwertbildung gefragt. Soll jeder Punkt erscheinen, wird [1] eingegeben, dann müssen aber Datenmengen größer als 640 auf mehrere Seiten verteilt werden. Andernfalls werden jeweils die Mittelwerte aus M Zahlen auf 1/M horizontal zusammengedrängt (dies besorgen später die Zeilen 51 bis 62).

Nach Anzeige der höchstmöglichen Punktzahl pro Seite und des größten und kleinsten Datenwerts wird eine der drei Plot-Arten erfragt. Schließlich erfolgt die Bestimmung des y-Maßstabs: Der oben angezeigte größte Datenwert wird durch diesen y-Faktor auf den Wert 200 (Fall 1) beziehungsweise 100 (Fall 3) transformiert. Im Fall 3 steht der Bereich 30 bis 375 zur Verfügung. Der Wert y kann eine beliebige Dezimalzahl sein. Wer bequem ist, überläßt es dem Programm, den bestmöglichen Faktor zu suchen. Das ist schon fast alles.

Es bleibt noch, den horizontalen Skalenfaktor festzulegen. Sind es wenige Daten (höchstens die halbe Bildschirmbreite), so wird automatische Streckung angeboten, andernfalls gibt man den x-Dehnungsfaktor (ganzzahlig) ein. Dieser hängt natürlich von der Gesamtdatenzahl und der eventuellen Mittelung ab. Daher wird die jetzt maximal anzeigbare Datenzahl pro Seite gemeldet und gegebenenfalls gefragt, welcher Bereich (Kanal K1 bis K2) geplottet werden soll. Man kann so nach dem Ausdrucken lange Datensätze auf mehrere Seiten verteilen (und diese später zusammenkleben).

Nun zur Skalierung: Je nach Mittelwertbildung M und x-Dehnung wird auf der unteren x-Achse eine passende Skalierung markiert (Zeilen 178 bis 180). Zur Orientierung werden dazu einige Ziffern angebracht, die für die häufigsten Fälle (Mittelung M gleich eins bis sechs und x-Dehnung ebenfalls zwischen eins bis sechs) ausreichen. Sie werden nach den Bedingungen der Zeilen 140 bis 158 plaziert und aus den Zeichensätzen (Zeile 191 bis Ende) geformt, die anstelle der klobigen Grafikmode-Ziffern treten und den Plot kaum stören. Trifft keine der Bedingungen zu, erscheint oben und unten die gleiche Punktskala. Han-

delt es sich sowohl um positive wie auch negative Zahlen (Modus 3), wird eine Mittelhorizontale gezeichnet, die als Nullinie (x-Achse) dient.
Die Jahresskala wird in den Zeilen 48 bis 53 mit der Jahresangabe des ersten Datenpunktes angewählt und schreibt 50er Jahresmarken (Zeilen 111 bis 117) und Jahrhundertziffern mit den Zeichensätzen.
Das Programm sucht den gewählten Datenbereich heraus, schreibt den Dateinamen in die rechte obere Ecke, berechnet die zu plottenden y-Werte und präsentiert sie auf dem Bildschirm entweder als Kurve (Zeile 82), als Histogramm (Zeile 84 und 85) oder als Punktfolge (Zeile 86). Anschließend erscheint ein Rahmen, und es werden Skalenmarken an die y-Achse geschrieben sowie an die x-Achse oben und unten, soweit nicht andere Skalierungen gewählt wurden. Jetzt ist auch der Augenblick zum Drucken mit der Taste [PrtScr] gekommen.
Um das Schirmbild (für den Ausdruck) frei von Texten zu halten, wird die folgende letzte Anfrage im Verborgenen gehalten: die Eingabe von [0],[1],[2],[3] oder [4]. Bei [0] kehrt das Programm zu Basic zurück. Bei [1] wird zuerst im Drucker das Papier zum nächsten Seitenbeginn transportiert und ein weiterer Datenbereich K1,K2 erwartet oder das Programm neu gestartet. Die Eingabe von [2] führt zu neuer Parameterwahl, und [3] zum Neustart des Programms ohne Papiervorschub. Schließlich leitet [4] zu MS-DOS zurück. Mit [CTRL-C] läßt sich natürlich jederzeit zu Basic zurückkehren.
Noch ein paar Tips: Vor Start von Kurplot muß »Graphics« geladen werden, bei Verwendung einer Hercules-Grafikkarte zusätzlich ein CGA-Emulator. Nach Änderung der maximalen x- beziehungsweise y-Punktzahl an allen Programmstellen kann man auch andere Grafikkarten verwenden, zum Beispiel Hercules mit 720 auf 348 Punkten. Des weiteren kann die an Mittelung und Dehnung angepaßte Skalierung, die einen großen Teil des Programmtextes einnimmt, ebenso weggelassen werden wie die Jahresskala. Dann sind die Zeilen 140 bis 158 und die Zeilen 1291 bis 211 zu streichen. Es erscheint daraufhin, ebenso an der oberen x-Achse, nur die reine Punktzahlskalierung in 50er Stufen.

```
Kurve (K) oder Histogramm (H) oder Punkte (P)  ? K
Filename, Einlesen A:HOHE.ORG

Datenzahl                    850
Plotten, Mittel aus M =      2

Plot pro Seite max.          1270
größter Datenwert =          375
kleinster Datenwert =        58

 1  positive Daten, Y-linear      0 < Y < 200
 2  positive Daten, Y-logar.     30 < Y < 375
 3    +/-  Daten  , Y-linear   -100 < Y < 100    1

Y-Höhe automatisch:    0, oder
Multiplikator  Y =     .5
X-Dehnungsfaktor  =    1

Plot pro Seite max.          1270
Daten von K1 bis K2          1,950
Jahres-Skalierung  J/N       J
erste Jahreszahl  (BC: negativ)   1881
```

Bild 2. Diese Eingaben führen in Verbindung mit einem entsprechenden Datensatz zu Bild 1

Wird Kurplot compiliert, so ist die Ablaufgeschwindigkeit deutlich höher. Bei zu kleinem Arbeitsspeicher kann die Dimensionierung der Variablen in Zeile 5 leicht verkleinert werden – es lassen sich dann allerdings nur kürzere Dateien verarbeiten.

(Wolfgang Gruhle/sw)

```
[VX4] 1   'Programmname: KURPLOT.BAS'
[7E4] 2   '(C) 1988 Everts & Hagedorn'
[Z84] 3   '          DOS International'
[ØG2] 4   DEFINT A-G,I-W
[BK3] 5   DIM X(6ØØØ),Y(6ØØØ),A(3Ø)
[211] 6   CLS
[GH9] 7   INPUT "Kurve (K) oder Histogramm (H) oder Punkte (P)  ";AD$
[9Z5] 8   LINE INPUT "Filename, Einlesen ";F$
[3R1] 9   KEY OFF
[R42] 1Ø  OPEN "I",1,F$
[VY1] 11  I=1:YMAX=Ø
[VU2] 12  WHILE NOT EOF(1)
[E22] 13  INPUT #1,S#
[KG1] 14  X(I)=S#
[EH3] 15  IF I=1 THEN YMIN=X(I)
[NP4] 16  IF X(I)>YMAX THEN YMAX=X(I)
[YP4] 17  IF X(I)<YMIN THEN YMIN=X(I)
[451] 18  I=I+1
[651] 19  WEND
[U41] 2Ø  N=I-1
[WX5] 21  PRINT:PRINT "Datenzahl                    ",N
[H21] 22  BEEP
[XH1] 23  CLOSE #1
[R66] 24  INPUT "Plotten, Mittel aus M =       ",MN
[CØ6] 25  PRINT "Plot pro Seite max. ",639*MN
[KTC] 26  PRINT "größter Datenwert =  ",YMAX:PRINT "kleinster Datenwert =  ",YMIN
[ZZ8] 27  PRINT:PRINT " 1  positive Daten, Y-linear      Ø < Y < 2ØØ "
[2U7] 28  PRINT " 2  positive Daten, Y-logar.      3Ø < Y < 375 "
[TU7] 29  INPUT " 3    +/-  Daten  , Y-linear   -1ØØ < Y < 1ØØ    ",E
[ZR3] 3Ø  PRINT:IF (E=2) GOTO 32
[XD6] 31  PRINT "Y-Höhe automatisch:    Ø, oder"
[BN5] 32  INPUT "Multiplikator  Y =      ",KM#
[AC4] 33  IF NOT (KM#=Ø) THEN GOTO 36
[ØV5] 34  IF E=3 THEN KM#=ABS(1ØØ/YMAX):GOTO 36
[1Y3] 35  IF E=1 THEN KM#=199/YMAX
[114] 36  IF N>MN*319 THEN GOTO 4Ø
[MS7] 37  PRINT:INPUT "automat. X-Streckung (J/N) ";AX$
[117] 38  IF AX$=CHR$(74) THEN M=MN*639\N ELSE GOTO 4Ø
[FG3] 39  L=1:J1=1:J2=N:GOTO 48
[1P6] 4Ø  INPUT "X-Dehnungsfaktor  =     ",MX:PRINT
[2G6] 41  PRINT "Plot pro Seite max. ",639*MN\MX
[SW4] 42  IF MX\MN<32767Ø\N THEN GOTO 44
```

```
[GLE] 43  PRINT N;"Daten und Mittel aus";MN;"geben max. mögliche X-Dehnung = ";(32767Ø\N)*MN-1:GOTO 4Ø
[CR8] 44  IF N<=639 AND 639>=(MX\MN)*N THEN L=1:J1=1:J2=N:GOTO 47
[UB6] 45  INPUT "Daten von K1 bis K2          ",K1,K2
[PF7] 46  J1=INT((K1+MN-1)\MN):J2=INT((K2+MN-1)\MN):L=K1
[A41] 47  M=MX
[HB6] 48  INPUT "Jahres-Skalierung  J/N       ",AS$
[PR3] 49  IF AS$=CHR$(78) GOTO 51
[687] 5Ø  INPUT "erste Jahreszahl  (BC: negativ)    ",JA
[2R2] 51  FOR I=J1 TO J2
[C95] 52  IF MN=1 THEN XM=KM#*X(I):GOTO 56
[M22] 53  XM=KM#*X(L)
[TR2] 54  FOR G=1 TO MN-1
[S54] 55  XM=XM+KM#*X(L+G):NEXT G
[RM5] 56  IF E=3 THEN Y(I)=1ØØ-XM\MN:GOTO 59
[U55] 57  IF E=2 THEN Y(I)=XM\MN:GOTO 59
[9P2] 58  Y(I)=199-XM\MN
[7C2] 59  L=L+MN:IL=I
[744] 6Ø  IF (L>N) AND (E=2) GOTO 64
[ZF2] 61  IF L>N GOTO 75
[WE1] 62  NEXT I
[GØ3] 63  IF NOT E=2 GOTO 75
[DR2] 64  FOR I=J1 TO J2
[754] 65  IF Y(I)<=Ø THEN Y(I)=.ØØØØ1
[YH3] 66  Y(I)=LOG(Y(I))/LOG(1Ø)
[R63] 67  Y(I)=199-181*Y(I)+267
[EF1] 68  NEXT I
[DR2] 69  FOR LY=4 TO 1Ø
[KN5] 7Ø  A(LY)=466-INT(181*LOG(1Ø*LY)/LOG(1Ø))
[8U1] 71  NEXT LY
[B15] 72  A(15)=466-INT(181*LOG(15Ø)/LOG(1Ø))
[C15] 73  A(2Ø)=466-INT(181*LOG(2ØØ)/LOG(1Ø))
[R15] 74  A(3Ø)=466-INT(181*LOG(3ØØ)/LOG(1Ø))
[ZD2] 75  CLS:SCREEN 2
[XL3] 76  LOCATE 2,67:PRINT F$;
[SL4] 77  PSET(M,Y(J1)):PSET(M-1,Y(J1))
[EU2] 78  FOR I=J1+1 TO J2
[9H3] 79  IF I>IL THEN GOTO 87
[8P3] 8Ø  IF AD$=CHR$(72) GOTO 84
[FP3] 81  IF AD$=CHR$(8Ø) GOTO 86
[UN3] 82  LINE -(M*(I+1-J1),Y(I))
[7M1] 83  GOTO 87
[7V3] 84  LINE -(M*(I+1-J1),Y(I-1))
[VY4] 85  LINE -(M*(I+1-J1),Y(I)):GOTO 87
[7R6] 86  PSET(M*(I+1-J1),Y(I)):PSET(1+M*(I+1-J1),Y(I))
```

```
[FF1] 87  NEXT I
[PG3] 88  LINE (Ø,Ø)-(639,199),,B
[9X3] 89  IF (E=1) OR (E=3) GOTO 97
[XK2] 9Ø  FOR I2=4 TO 1Ø
[AX5] 91  FOR I1=1 TO 2:PSET(I1,A(I2)):NEXT I1
[9H7] 92  FOR I1=637 TO 638:PSET (I1,A(I2)):NEXT I1:NEX
      T I2
[7Y9] 93  PSET(1,A(15)):PSET(3,A(5)):PSET(3,A(1Ø)):PSET
      (4,A(1Ø)):PSET(1,A(2Ø))
[L48] 94  PSET(1,A(3Ø)):PSET(2,A(2Ø)):PSET(2,A(3Ø)):PSE
      T(635,A(1Ø))
[MZ8] 95  PSET(638,A(15)):PSET(638,A(2Ø)):PSET(638,A(3Ø
      )):PSET(637,A(2Ø))
[Z68] 96  PSET(637,A(3Ø)):PSET(636,A(5)):PSET(636,A(1Ø)
      ):GOTO 1Ø7
[C82] 97  FOR I=1 TO 2
[EZ3] 98  FOR I1=2Ø TO 18Ø STEP 2Ø
[ZX1] 99  PSET(I,I1)
[4U2] 1ØØ NEXT I1:NEXT I
[HP3] 1Ø1 PSET(3,1ØØ):PSET(4,1ØØ)
[1T2] 1Ø2 IF (E=3) GOTO 1Ø7
[6V2] 1Ø3 FOR I=637 TO 638
[414] 1Ø4 FOR J=2Ø TO 18Ø STEP 2Ø
[N74] 1Ø5 PSET(I,J):NEXT J:NEXT I
[394] 1Ø6 PSET(636,1ØØ):PSET(635,1ØØ)
[824] 1Ø7    IF AS$=CHR$(74) GOTO 111
[T14] 1Ø8 FOR I=5Ø TO 6ØØ STEP 5Ø
[CT2] 1Ø9 PSET(I,1):NEXT
[1R6] 11Ø PSET(25Ø,2):PSET(5ØØ,2):PSET(5ØØ,3):GOTO 159
[ØV7] 111 IF JA<Ø THEN Q5=(JA\5Ø)*5Ø:Q1Ø=(JA\1ØØ)*1ØØ:G
      OTO 113
[CZ4] 112 Q5=(1+JA\5Ø)*5Ø:Q1Ø=(1+JA\1ØØ)*1ØØ
[4A2] 113 FOR B=1 TO N
[6V3] 114 R5=(Q5-JA+5Ø*(B-1))*M\MN
[VY2] 115 IF R5>639 GOTO 118
[HØ2] 116 PSET(R5,1)
[G81] 117 NEXT
[SA2] 118 FOR B=1 TO N
[C46] 119 R1Ø=((Q1Ø-JA+1ØØ*(B-1))*M)\MN:PSET(R1Ø,2)
[NS4] 12Ø IF R1Ø>639 GOTO 121 ELSE NEXT
[1D3] 121 R1Ø=((Q1Ø-JA)*M)\MN
[BK3] 122 JAA=ABS(Q1Ø):JAB=Q1Ø
[2T7] 123 IF(MN>1 AND M>1) THEN FS=(1ØØ*M)/MN ELSE FS=1
      ØØ*M
[484] 124 FOR F=R1Ø TO 639 STEP FS
[W83] 125 IF F+17>639 GOTO 159
[NK8] 126 JZ1=JAA\1ØØØ:JZ2=(JAA\1ØØ)-JZ1*1Ø:JZ3=JAA-(JA
      A\1ØØ)*1ØØ
[6P8] 127 IF (JZ2=Ø AND NOT JZ1=Ø) THEN FZ=F-2:GOSUB 14
      Ø:GOTO 13Ø
[WD1] 128 FZ=F-2
[HR6] 129 ON JZ2 GOSUB 142,143,145,147,149,151,153,155,
      157
[WY2] 13Ø FZ=F+8:GOSUB 14Ø
[EG4] 131 IF (JZ1=Ø AND JZ2=Ø) GOTO 133
[EØ3] 132 FZ=F+17:GOSUB 14Ø
[8W2] 133 IF JZ1=Ø GOTO 137
[9S4] 134 IF JZ2=1 THEN FZ=F-8:GOTO 136
[NE5] 135 IF JZ2=7 THEN FZ=F-1Ø ELSE FZ=F-12
[5R6] 136 ON JZ1 GOSUB 142,143,145,147,149,151,153,155,
      157
[7Ø7] 137 IF M>1 THEN JAB=JAB+1ØØ ELSE JAB=JAB+1ØØ*MN
[RE2] 138 JAA=ABS(JAB)
[ØL2] 139 NEXT:GOTO 159
[NX6] 14Ø FOR I=4 TO 6:PSET(FZ-6,I):PSET(FZ,I):NEXT
[T19] 141 FOR I=1 TO 2:PSET(FZ-2*I,3):PSET(FZ-2*I,7):NE
      XT:RETURN
[498] 142 FOR I=3 TO 7:PSET(FZ,I):NEXT:PSET(FZ-2,4):RET
      URN
[U1B] 143 FOR I=Ø TO 3:PSET(FZ-2*I,7):NEXT:PSET(FZ,4):P
      SET(FZ-2,3):PSET(FZ-2,5)
[DG7] 144 PSET(FZ-4,3):PSET(FZ-4,6):PSET(FZ-6,4):RETURN
[N9B] 145 FOR I=1 TO 3:PSET(FZ-2*I,3):PSET(FZ-2*I,7):NE
      XT:PSET(FZ,4):PSET(FZ,6)
[ZN5] 146 PSET(FZ-2,5):PSET(FZ-4,5):RETURN
[YY9] 147 FOR I=4 TO 7:PSET(FZ-2,I):NEXT:FOR I=3 TO 5:P
      SET(FZ-6,I):NEXT
[1D5] 148 PSET(FZ-4,5):PSET(FZ,5):RETURN
[F1A] 149 FOR I=1 TO 3:FOR K=3 TO 7 STEP 2:PSET(FZ-2*I,
      K):NEXT K:NEXT I
[ØD5] 15Ø PSET(FZ,6):PSET(FZ-6,4):RETURN
[U8B] 151 FOR I= 4 TO 6:PSET(FZ-6,I):NEXT:PSET(FZ-2,3):
      PSET(FZ-4,3):PSET(FZ-2,5)
[WD9] 152 PSET(FZ-4,5):PSET(FZ,6):PSET(FZ-2,7):PSET(FZ-
      4,7):RETURN
[AU9] 153 PSET(FZ,3):PSET(FZ,4):PSET(FZ-2,3):PSET(FZ-4,
      3):PSET(FZ-2,5)
[4P5] 154 PSET(FZ-4,6):PSET(FZ-4,7):RETURN
[11A] 155 FOR I=1 TO 2:FOR K=3 TO 7 STEP 2:PSET(FZ-2*I,
      K):NEXT K:NEXT I
[Ø49] 156 PSET(FZ,4):PSET(FZ,6):PSET(FZ-6,4):PSET(FZ-6,
      6):RETURN
[3ØA] 157 FOR I=4 TO 6:PSET(FZ,I):NEXT:FOR I=1 TO 2:FOR
       K=3 TO 7 STEP 2
[VC8] 158 PSET(FZ-2*I,K):NEXT K:NEXT I:PSET(FZ-6,4):RET
      URN
[LRB] 159 IF M=MN THEN HM=5Ø:F=2ØØ:GOSUB 195:F=4ØØ:GOSU
      B 2Ø1:F=6ØØ:GOSUB 2Ø6:GOTO 179
[BGD] 16Ø IF MN=2*M OR M=5*MN THEN HM=5Ø:F=25Ø:GOSUB 2Ø
      3:F=496:GOSUB 194:F=5Ø2:GOSUB 191:GOTO 179
[XY9] 161 IF 2*M=3*MN THEN HM=75:F=3ØØ:GOSUB 195:F=6ØØ:
      GOSUB 2Ø1:GOTO 179
[9GD] 162 IF MN=5*M OR M=2*MN THEN HM=4Ø:F=2ØØ:GOSUB 19
      4:F=4ØØ:GOSUB 195:F=6ØØ:GOSUB 198:GOTO 179
[X4H] 163 IF MN=3*M THEN HM=33.33:F=167:GOSUB 2Ø3:F=329
      :GOSUB 194:F=335:GOSUB 191:F=496:GOSUB 194:F=5Ø2:
      GOSUB 2Ø3:GOTO 179
[V9A] 164 IF 3*M=2*MN THEN HM=66.67:F=267:GOSUB 2Ø1:F=5
      33:GOSUB 2Ø9:GOTO 179
[Y9A] 165 IF 3*M=4*MN THEN HM=66.67:F=267:GOSUB 195:F=5
      33:GOSUB 2Ø1:GOTO 179
[KMC] 166 IF 3*M=5*MN THEN HM=33.33:F=167:GOSUB 194:F=3
      33:GOSUB 195:F=5ØØ:GOSUB 198:GOTO 179
[KF9] 167 IF M=3*MN THEN HM=6Ø:F=3ØØ:GOSUB 194:F=6ØØ:GO
      SUB 195:GOTO 179
[8HK] 168 IF 5*M=2*MN OR M=4*MN THEN HM=4Ø:F=2ØØ:GOSUB
      2Ø3:F=396:GOSUB 194:F=4Ø2:GOSUB 191:F=596:GOSUB 1
      94:F=6Ø2:GOSUB 2Ø3:GOTO 179
[45B] 169 IF MN=4*M OR 2*M=5*MN THEN HM=5Ø:F=25Ø:GOSUB
      194:F=5ØØ:GOSUB 195:GOTO 179
[K9C] 17Ø IF 5*M=3*MN THEN HM=6Ø:F=3ØØ:GOSUB 2Ø3:F=596:
      GOSUB 194:F=6Ø2:GOSUB 191:GOTO 179
[G9C] 171 IF 5*M=4*MN THEN HM=4Ø:F=16Ø:GOSUB 195:F=32Ø:
      GOSUB 2Ø1:F=48Ø:GOSUB 2Ø6:GOTO 179
[PØA] 172 IF 4*M=3*MN THEN HM=75:F=225:GOSUB 198:F=45Ø:
      GOSUB 2Ø6:GOTO 179
[G6A] 173 IF 4*M=5*MN THEN HM=62.5:F=25Ø:GOSUB 195:F=5Ø
      Ø:GOSUB 2Ø1:GOTO 179
[XCC] 174 IF MN=6*M THEN HM=33.33:F=167:GOSUB 194:F=333
      :GOSUB 195:F=5ØØ:GOSUB 198:GOTO 179
[ØXB] 175 IF M=6*MN THEN HM=6Ø:F=3ØØ:GOSUB 2Ø3:F=596:GO
      SUB 194:F=6Ø2:GOSUB 191:GOTO 179
[GØA] 176 IF 5*M=6*MN THEN HM=6Ø:F=24Ø:GOSUB 195:F=48Ø:
      GOSUB 2Ø1:GOTO 179
[T8A] 177 IF 6*M=5*MN THEN HM=83.33:F=25Ø:GOSUB 198:F=5
      ØØ:GOSUB 2Ø6:GOTO 179
[781] 178 HM=5Ø
[FB4] 179 FOR H=HM TO 63Ø STEP HM
[P34] 18Ø PSET (INT(H),198):NEXT H
[RC3] 181 IF NOT (E=3) THEN 183
[383] 182 LINE (Ø,1ØØ)-(639,1ØØ)
[AZC] 183 INPUT "",P   'Eingabe Ø=Ende,1=Neustart K1-K2
      ,2=neue Parameter,3=Neustart,4=DOS
[4E3] 184 IF P=Ø THEN GOTO 19Ø
[HD5] 185 IF P=2 THEN CLS:SCREEN Ø:GOTO 31
[TC3] 186 IF P=4 THEN SYSTEM
[7N4] 187 IF P=3 THEN SCREEN Ø:GOTO 6
[ØV9] 188 IF P=1 AND (639*MN\M<N) THEN LPRINT CHR$(12):
      SCREEN Ø:GOTO 45
[H17] 189 IF P=1 THEN LPRINT CHR$(12):SCREEN Ø:GOTO 6
[H31] 19Ø END
[MS9] 191 PSET(F-2,192):PSET(F-2,193):PSET(F-2,194):PSE
      T(F,191):PSET(F,195)
[86A] 192 PSET(F+2,191):PSET(F+2,195):PSET(F+4,192):PSE
      T(F+4,193):PSET(F+4,194)
[ZM1] 193 RETURN
[LV8] 194 PSET(F-2,192):FOR I=191 TO 195:PSET(F,I):NEXT
       I:RETURN
[UK9] 195 PSET(F-2,192):PSET(F-2,195):PSET(F,191):PSET(
      F,194):PSET(F,195)
[H6A] 196 PSET(F+2,191):PSET(F+2,193):PSET(F+2,195):PSE
      T(F+4,192):PSET(F+4,195)
[FN1] 197 RETURN
[2L9] 198 PSET(F-2,191):PSET(F-2,195):PSET(F,191):PSET(
      F,193):PSET(F,195)
[W6A] 199 PSET(F+2,191):PSET(F+2,193):PSET(F+2,195):PSE
      T(F+4,192):PSET(F+4,194)
[UL1] 2ØØ RETURN
[NØA] 2Ø1 PSET(F-4,191):PSET(F-4,192):PSET(F-4,193):PSE
      T(F-2,193):PSET(F,192)
[TV8] 2Ø2 PSET(F,193):PSET(F,194):PSET(F,195):PSET(F+2,
      193):RETURN
[PØA] 2Ø3 PSET(F-2,191):PSET(F-2,192):PSET(F-2,193):PSE
      T(F-2,195):PSET(F,191)
[HP9] 2Ø4 PSET(F,193):PSET(F,195):PSET(F+2,191):PSET(F+
      2,193):PSET(F+2,195)
[RN3] 2Ø5 PSET(F+3,194):RETURN
[DS9] 2Ø6 PSET(F-2,192):PSET(F-2,193):PSET(F-2,194):PSE
      T(F,191):PSET(F,193)
[ZØA] 2Ø7 PSET(F,195):PSET(F+2,191):PSET(F+2,193):PSET(
      F+2,195):PSET(F+3,194)
[UM1] 2Ø8 RETURN
[BK9] 2Ø9 PSET(F-2,192):PSET(F-2,194):PSET(F,191):PSET(
      F,193):PSET(F,195)
[25A] 21Ø PSET(F+2,191):PSET(F+2,193):PSET(F+2,195):PSE
      T(F+4,192):PSET(F+4,194)
[1M1] 211 RETURN
```

Listing. Mit dem Basic-Programm »kurplot.bas« drucken Sie ohne viel Aufhebens einfache Diagramme aus

Eine abstrakte Maschine

Computersysteme werden nach wie vor von vielen als komplizierte, unüberschaubare technische Gebilde angesehen: Man programmiert sie zwar, aber was sich innerhalb der »Black Box« abspielt, bleibt ein Buch mit sieben Siegeln. Betrachtet man jedoch einzelne überschaubare Gruppen, so wächst sehr schnell das Verständnis für diese Thematik. Die Simulation eines einfachen Computersystems hilft Ihnen dabei.

Im folgenden wird ein Pascal-Programm vorgestellt, das ein sehr einfaches Computersystem (»SEC«) definiert. Den Kern des Systems bildet eine abstrakte Maschine (»MAC«) mit den Teilen Prozessor, Speichereinheit und Ein-/Ausgabe-Einheit.

Ausgehend von der grundlegenden Beschreibung der abstrakten Maschine wird zunächst ein Programm zur Realisierung dieser Maschine definiert, um dann zu einem System erweitert zu werden, das die Eingabe und den Start von Maschinenprogrammen in hexadezimaler Darstellung erlaubt.

Zum Ausbau des MAC-Systems wird ein Assembler vorgeschlagen und in das System integriert, so daß Programme sowohl im Maschinencode als auch als Assembler-Programmtexte eingegeben werden können. Das vorgestellte System wurde so einfach gewählt, um die Möglichkeit der Maschinendarstellung klar zu zeigen und für Erweiterungen weitestgehend offen zu sein.

Mit dem vorgeschlagenen Programm als Grundlage sind die verschiedensten Erweiterungen und Änderungen denkbar:

- Erweiterung des Maschinenbefehlssatzes gemäß den Bedürfnissen einer bestimmten Anwendung.
- Unterteilung der vorgestellten Einheiten in Subeinheiten.
- Darstellung der prozessorinternen Abläufe auf Bitebene.
- Umgestaltung des Assemblers nach den bekannten Techniken des Assemblerbaus.
- Emulation von realen Maschinen.

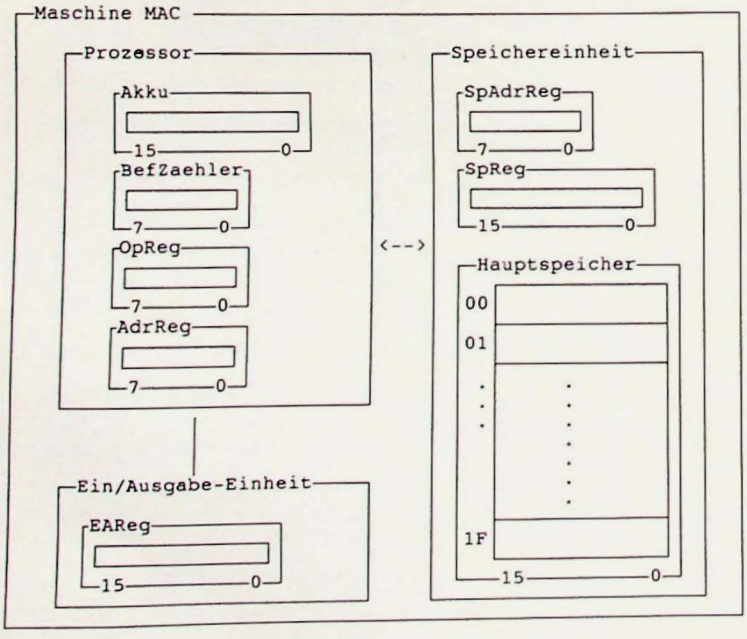


Bild 1. Grafische Darstellung der informalen Beschreibung der abstrakten Maschine MAC

Die abstrakte Maschine (MAC) besteht aus den Teilen Prozessor, Speichereinheit und Ein-/Ausgabe-Einheit. Der Befehlsvorrat der Maschine enthält je einen Befehl zum Laden und Speichern aus dem beziehungsweise in den Hauptspeicher (»LAD«, »SPE«), einen Befehl zum Laden einer Konstanten (»NUM«), zwei arithmetische Befehle für die Addition und die Subtraktion (»ADD«, »SUB«), zwei Sprungbefehle für den unbedingten und den bedingten Sprung (»SPR«, »SPN«) und je einen Befehl für die Ein- und Ausgabe (»EIN«, »AUS«). Der Befehl Halt (»HLT«) zeigt das Programmende an.

Aus einem Prozessor, einer Speichereinheit und einer Ein-/Ausgabe-Einheit wird eine abstrakte Maschine entwickelt

Das Befehlsformat der Maschine wird so vereinbart, daß ein Maschinenwort (16 Bit) den Operationsteil (8 Bit) und den Adreßteil (8 Bit) eines Befehls enthält. Da auch der Hauptspeicher eine Breite von 16 Bit besitzt, passen der Operationscode und die Datenadresse eines Befehls in eine Speicherzelle.

Der Prozessor enthält die vier Register Akkumulator (»Akku«, 16 Bit), Befehlszähler (»BefZae«, 8 Bit), Operationsregister (»OpReg«, 8 Bit) und Adreßregister (»AdrReg«, 8 Bit). Die Verarbeitung von im Hauptspeicher stehenden Befehlen (beziehungsweise Befehlsfolgen, also Programmen) durch den Prozessor erfolgt gemäß seinem Maschinenzyklus: Nach dem Lesen des Inhalts einer Speicherzelle aus dem Hauptspeicher wird dieser Inhalt auf das Operationsregister und das Adreßregister verteilt. Dem Inhalt des Operationsregisters entsprechend wird dann ein bestimmter Maschinenbefehl ausgeführt. Eventuell nötige Daten werden dabei über das Adreßregister adressiert. Dadurch lassen sich Programmdaten ebenfalls aus dem Hauptspeicher holen oder dort speichern.

Der Prozessor kommuniziert mit den beiden ihn umgebenden Einheiten über die Register Speicherregister (»SpReg«, 16 Bit) für die Speichereinheit und Ein-/Ausgaberegister (»EAReg«, 8 Bit) für die Ein-/Ausgabe-Einheit. Er spricht die beiden Einheiten über die vier Aktionen Hauptspeicher-Lesen (»HSpLesen«), Hauptspeicher-Schreiben (»HSpSchreiben«), Eingabe (»eing«) und Ausgabe (»ausg«) an und löst die gewünschte Aktion aus.

Die Speichereinheit besteht aus dem Hauptspeicher (»HSp«), der eine Breite von 16 Bit und eine Größe (Länge) von 32 Worten hat, und den beiden Registern Speicherregister (»SpReg«, 16 Bit) und Speicheradreßregister (»SpAdrReg«, 8 Bit). Mit den Operationen Schreiben (»HSpSchreiben«) und Lesen (»HSpLesen«) werden Speicherinhalte in den Hauptspeicher geschrieben oder aus ihm gelesen. Die Adresse der gewünschten Speicherzelle ist dabei anzugeben. Ein zu schreibender Speicherinhalt muß vor dem Schreiben im Speicherregister stehen. Ebenso fällt ein gelesener Inhalt im Speicherregister an. Das Speicherregister ist also der Kanal für die Datenübertragung aus der und in die Speichereinheit.

Die Kommunikation der Maschine mit der Umgebung geschieht über die Ein-/Ausgabe-Einheit. Hier greift der Prozessor auf den Inhalt des Ein-/Ausgabe-Registers (»EAReg«, 16 Bit) zu – das einzige Register der Ein-/Ausgabe-Einheit – und löst Aktionen zur Eingabe (von der Tastatur) und Ausgabe (auf den Bildschirm) aus. Bild 1 gibt

Ihnen eine grafische Darstellung der informalen Beschreibung der abstrakten Maschine.
Die bisherige Beschreibung von MAC ist keine formale Definition. Dafür müßte ein geeigneter Formalismus überlegt werden, mit dem eine exakte Definition erst möglich würde. Dies soll in diesem Beitrag jedoch nicht geschehen, da die Umsetzung der Beschreibung in ein Pascal-Programm einer formalen Definition entspricht. Beachten Sie also bitte, daß die informale Beschreibung von MAC durch den Formalismus einer Programmiersprache formal gefaßt wird – mit dem Nebeneffekt, daß die formale Definition zugleich eine Realisierung von MAC darstellt.

Ein Pascal-Programm macht aus einer informalen Beschreibung eine formale Definition des MAC-Systems

Gemäß der Beschreibung von MAC, die drei getrennte Einheiten angibt, müssen Sie nun drei Programm-Module (Units) entwickeln: jeweils eines für den Prozessor, die Speichereinheit und die Ein-/Ausgabe-Einheit. Das erste Modul soll die Speichereinheit definieren.
Die Speichereinheit (Listing 1) besteht aus den beiden Registern Speicherregister und Speicheradreßregister und dem Hauptspeicher. Sie läßt die beiden Operationen Hauptspeicher-Lesen und Hauptspeicher-Schreiben zu. Der Hauptspeicher ist eine Aneinanderreihung von Speicherzellen, von denen jede eine eigene Nummer (die Adresse) hat und 16 Bit lang ist. Der geeignete Datentyp für den Hauptspeicher ist damit das »Array«, für die einzelnen Speicherzellen wird man den Typ »Word« wählen. Da sich der Inhalt des Hauptspeichers von Programm zu Programm und auch während der Ausführung eines Programms ändert, wird der Hauptspeicher als Variable deklariert:

```
var
HSp: array[$00..$1F] of word;
```

Ebenso vereinbaren Sie die Register als Variablen:

```
var
SpReg: word;
SpAdrReg: byte;
```

Nun die beiden Operationen: Sie werden vom Prozessor ausgelöst und veranlassen selbst gewisse Aktionen. Prozeduren haben die Eigenschaft, aufrufbar zu sein und Anweisungen auszuführen. Definieren Sie daher die Operationen als Prozeduren, wie dies in Listing 1 (Zeilen 20 bis 23 und 25 bis 28) geschehen ist.

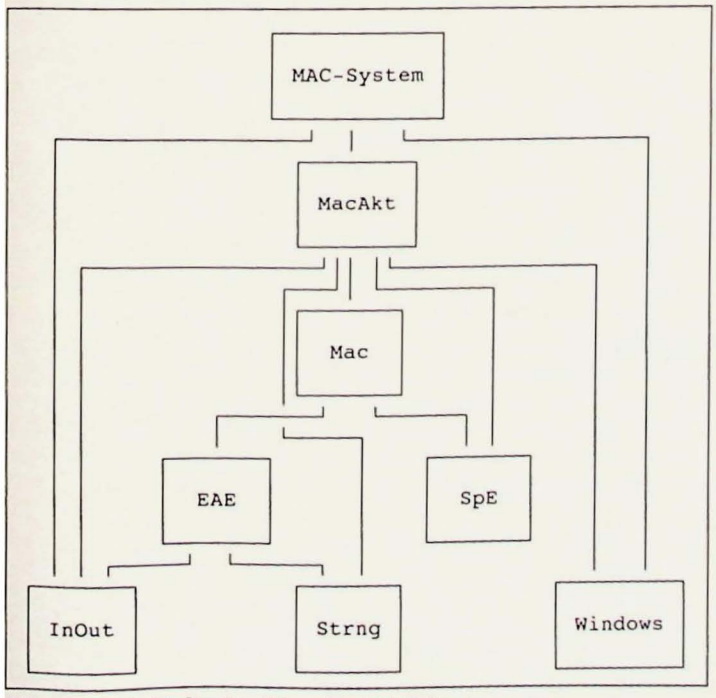


Bild 2. So sieht die Struktur des MAC-Systems aus

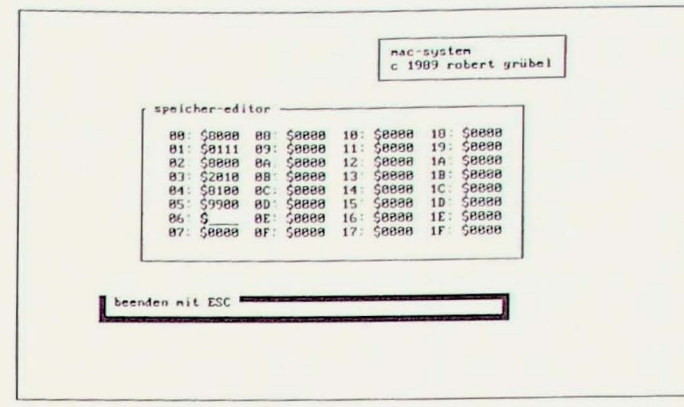


Bild 3. Bildschirmaufbau beim Eintragen eines Programms in den Speicher

Beachten Sie bitte, daß der Hauptspeicher und das Speicheradreßregister im Implementierungsteil der Unit deklariert werden und man deshalb nicht von außen auf sie zugreifen kann. Diese Abschirmung entspricht direkt der Forderung aus der Beschreibung der Speichereinheit, die klar besagt, daß auf die Speichereinheit nur über das Speicherregister und die Schreib- und Lese-Operationen zugegriffen werden darf. Würden Sie die Speichereinheit nicht als Unit, sondern einfach als Programmteil schreiben, dann wäre diese Forderung nicht einzuhalten: Der Hauptspeicher wäre als globale Variable im ganzen Programm bekannt, und Sie könnten direkt auf die einzelnen Speicherzellen (Array-Komponenten) zugreifen. Die Definition der Speichereinheit als Unit erlaubt also die Einhaltung der Schnittstellen-Definition der Speichereinheit auch im Programm.

Die Definition der Ein-/Ausgabe-Einheit ist einfach: Sie ist in Listing 2 wiedergegeben.

Nun wartet der Prozessor auf seine Realisierung. Die Prozessorbefehle werden als Konstanten definiert (Listing 3, Zeilen 31 bis 37), die das jeweilige Befehlskürzel mit dem Befehlscode gleichsetzen. Für die Prozessorregister verwenden Sie wieder Variablen. Neben den Registervariablen werden zwei Boolesche (logische) Variablen deklariert: Die Varibale »halt«, die das Erreichen des Programmendes anzeigt und eine Variable »fehler« für eventuell aufgetretene Fehler.

Mit Konstanten, Variablen und einigen Programmzeilen wird ein Prozessor realisiert

Der Maschinenzyklus des Prozessors ist eine Programmschleife, die bis zum Erreichen des Halt-Befehls (oder eines Fehlers) jeweils einen Speicherzelleninhalt liest, ihn auf das Operations- und das Adreßregister verteilt und die Ausführung des Befehls veranlaßt. Eine Umsetzung in Pascal sehen Sie in Listing 3 (Zeile 80 bis 86).

Das Lesen der richtigen Speicherzelle steuert der Befehlszähler. Er zeigt auf die Adresse des jeweils nächsten Befehls. Für die Befehlsausführung wird ein Prozeduraufruf verwendet. Die Prozedur »BefehlsAusfuehrung« wird gemäß dem Inhalt des Parameters »OpReg« eine gewisse Aktion auslösen. »Gemäß dem Inhalt von OpReg« bedeutet in Pascal

```
case OpReg of
```

und findet sich in Listing 3 (Zeile 41 bis 76) wieder. Die besprochenen Teile werden in der Unit »mac.pas« (Listing 3) gesammelt.

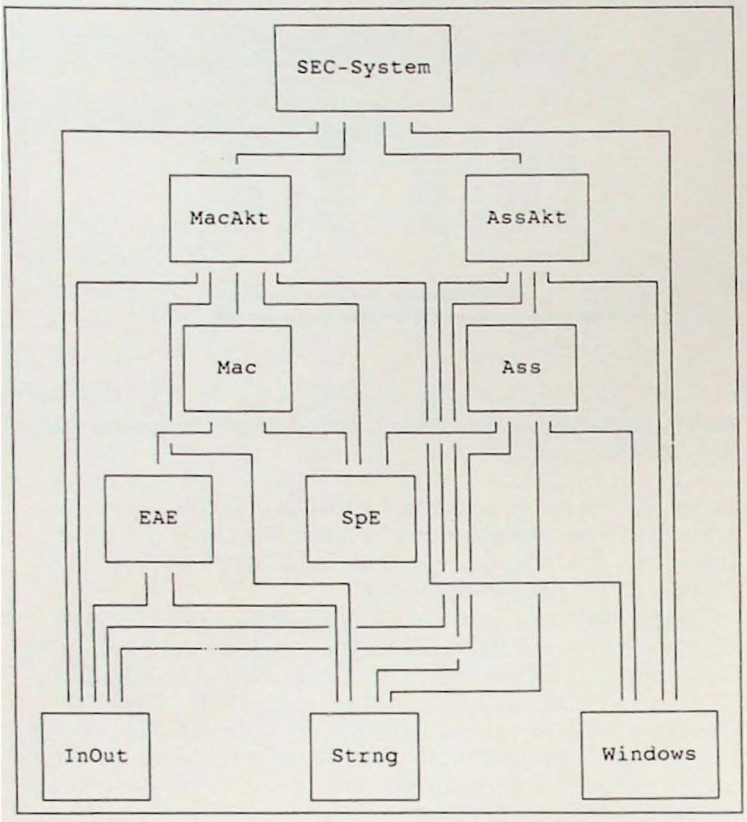


Bild 4. Darstellung der Strukturzusammenhänge des SEC-Systems

Beachten Sie bitte, daß der Prozessor eine ausführbare Einheit darstellt und deshalb als Prozedur definiert werden muß. Ein Maschinenlauf von MAC wird durch einen Aufruf der Prozedur »Prozessor« gestartet. »Prozessor« steht deshalb im Interface-Teil der Unit. Auch der Befehlszähler muß im Interface-Teil stehen, weil er vor der Programmausführung von außen eingestellt wird.

Nun wollen Sie natürlich mit MAC arbeiten. Sie möchten Programme in den Speicher schreiben und sie ausführen lassen. Ein kleines Testprogramm (Listing 4) soll erst einmal prüfen, ob MAC überhaupt funktioniert.

Zum Test wird ein Additionsprogramm in den Speicher von MAC geschrieben und gestartet. Für das Schreiben in den Speicher verwenden Sie die Operation »HSpSchreiben« der Speichereinheit, die den zu schreibenden Inhalt im Speicherregister erwartet. Der Aufruf von »Prozessor« löst den Start von MAC aus. Vor dem Start muß der Befehlszähler auf der Anfangsadresse des Programms stehen.

Um mit MAC zu arbeiten, müssen Sie zumindest Programme in den Speicher schreiben, sie ändern und starten können. Damit sich diese Aktionen komfortabel ausführen lassen, bietet es sich an, ein Menü zu verwenden. Eine Steuerung für ein solches Menü stellen die Zeilen 26 bis 43 in Listing 5 dar.

Eine Menüoberfläche erlaubt ein komfortables Arbeiten mit dem abstrakten Computer

Zu den bereits vorgeschlagenen ist hier noch eine Aktion »RegisterEditor« hinzugekommen, die die Inhalte der Register anzeigt und zur Änderung breitstellt. Um den »SpeicherEditor« und den »RegisterEditor« komfortabel programmieren und bedienen zu können, sind – gegenüber dem Turbo-Pascal-Standard – erweiterte Eingaberoutinen nötig. Für die Definition der Oberfläche des Bedienprogramms benötigen Sie außerdem minimale Fensteroperationen. Die geforderten Routinen und Operationen stellen die beiden Units »inout.pas« (Listing 6) und »windows.pas« (Listing 7) zur Verfügung. Eine weitere Unit »strng.pas« (Listing 8) stellt Routionen für die Stringbearbeitung bereit, zum Beispiel zur Umwandlung von Byte- oder Word-Werten in Strings, so daß alle Ein- und Ausgaben von MAC in hexadezimaler Form erfolgen.

Alle für die Menüsteuerung vorgeschlagenen Aktionen finden Sie in der Unit »macakt.pas« (Listing 9) wieder.

Noch einige Anmerkungen zu den Prozeduren »SpeicherEditor« und »RegisterEditor«:

- In beiden Prozeduren wird zunächst der Speicherinhalt, der sich ändern läßt, vollständig auf dem Bildschirm ausgegeben.
- Dann werden in einer Schleife die einzelnen Zeilen (beim Speicher die einzelnen Speicherzelleninhalte, bei den Registern die einzelnen Registerinhalte) zur Änderung bereitgestellt.
- Jeweils nach der Änderung einer Zeile wird die Zeilennummer auf die nächste zu ändernde Zeile erhöht oder entsprechend einer eventuell betätigten Sondertaste neu eingestellt.
- Die Prozedur »RegisterEditor« ruft eine Prozedur »RegisterWerte« auf. Diese findet sich – des Komforts wegen – auch in der Unit »mac.pas« wieder.

Bild 2 gibt Ihnen einen Überblick über die Struktur des MAC-Systems, und Bild 3 zeigt ein Beispiel für den Bildschirmaufbau.

Wie wäre es nun, wenn Sie MAC-Programme in folgender Form eingeben könnten?

```
EIN
SPE 10
EIN
ADD 10
AUS
HLT
```

Eine Form, die Sie von Assemblerprogrammen her kennen. Damit ein solches Assemblerprogramm wieder unter MAC läuft, muß jedes Befehlskürzel in seinen Code umgewandelt werden. Die Umwandlung ist eine einfache (Eins-zu-eins-) Umsetzung von Stringwerten in Zahlenwerte und erfolgt natürlich durch ein Programm: den Assembler.

Ein Assembler erleichtert Ihnen das Schreiben von Programmen für das MAC-System

Die Grundaufgabe des Assemblers ist die Umwandlung eines Programmtextes in einen Programmcode. Insbesondere bedeutet dies, daß zu bestimmten Zeichenketten bestimmte Zahlenwerte anzugeben sind. Im vorliegenden Fall müssen zu den jeweiligen Befehlskürzeln (Zeichenketten wie »EIN«, »ADD« und so weiter) bestimmte Befehlscodes gesucht und zu den Adreßangaben (Zeichenkombinationen wie »10«, »11« und so weiter) die entsprechenden Zahlenwerte ermittelt werden.

Da die Assembler-Eingangsdaten offensichtlich Programmtexte sind, wird man sie als in Zeilen unterteilte Zeichenketten verstehen – also die Textdateien von Turbo Pascal. Um aber während der Assemblierung nicht immer auf eine Datei (und damit ein externes Speichermedium) zugreifen zu müssen, wird hier ein Puffer für den zu assemblierenden Programmtext verwendet. Dafür definieren Sie einen Variablentyp »tTextPuffer« (um den Puffer auch als Parameter übergeben zu können):

```
type
tTextPuffer = array[0..15] of string[6];
```

Daß der Puffer gemäß dieser Definition eine maximale Programmlänge von nur 16 Zeilen zuläßt, ist eine eher angenehme Einschränkung. Dadurch kommt man nicht auf die Idee, MAC-Programme mit einer Länge zu schreiben, für die das System eigentlich nicht gedacht ist. Die maximale Breite einer Programmzeile beträgt sechs Zeichen

(drei für das Befehlskürzel, eine Leerstelle und eine zweistellige Zahl); auch das schlägt sich in der Puffertyp-Definition nieder.
Zum Vorgang der Assemblierung: Es soll hier kein klassischer Assembler vorgestellt werden. Vielmehr wird von einem (weitgehend) richtigen Programmtext ausgegangen, dessen Zeilen formatiert sind; das heißt: Befehl und Adresse stehen an bestimmten Stellen der Textzeile. Damit darf eine Fehlerbehandlung fehlen, und der Quelltext muß nicht zeichenweise bearbeitet werden. Geht man für den Assembler von einer Prozedur »Assembler« mit dem Prozedurkopf

```
procedure Assembler(var TextPuffer: tTextPuffer);
```

aus, dann kann der Assembler den Text zeilenweise aus dem Textpuffer lesen, bis das Kürzel für den Haltbefehl auftritt (Listing 10, Zeile 59 bis 85), und zur getrennten Codierung von Befehl und Adresse jeweils einen Teilstring für den Befehl und die Adresse bilden (Zeile 63 bis 68).
Zur Ermittlung der Adresse genügt es, den Adreß-String in sein Zahlenäquivalent umzuwandeln. Dafür verwenden Sie die Standard-Stringprozedur »val« (»AdrString« wird dabei als eine hexadezimale Zahlendarstellung interpretiert):

```
val('$'+ AdrString, Adresse, fcode);
```

Die Codierung des Befehls-Strings ist komplizierter: Der String und seine Codenummer haben eigentlich nichts miteinander zu tun, die Zuordnung ist willkürlich. Für die Codierung bietet sich ein Array an, in das Sie alle zulässigen Befehlskürzel schreiben; und zwar so, daß ihr Code jeweils mit ihrem Index übereinstimmt. Dann müssen Sie nur das Array mit dem aktuellen Befehls-String durchsuchen, und falls das Array den String enthält, ist der entsprechende Index identisch mit dem Befehlscode.

Ein schneller Zugriff des Assemblers auf die einzelnen Maschinenbefehle des Prozessors ist erwünscht

Das Array »MacBefehle« definieren Sie als typisierte Konstante (Listing 10, Zeile 19 bis 41). Daß dabei Platz verschwendet wird, gleicht die einfache Änderbarkeit des Assemblers für zusätzliche Befehle wieder aus. Das Suchen nach einem bestimmten Befehlskürzel im Array »MacBefehle« geschieht durch eine einfache While-Schleife (Zeile 69 bis 72).
Nun liegen der Befehlscode und gegebenenfalls eine Adresse als Byte-Werte vor. Zum Ablegen des erzeugten Codes bietet sich natürlich der Hauptspeicher von MAC an; dort werden Sie das Programm ausprobieren wollen. Dazu müssen Sie die beiden Byte-Werte in das Speicherregister schreiben, damit diese durch einen Aufruf von »HSpSchreiben« in den Speicher gelangen. Das Zusammenfügen der beiden Byte-Werte (zu einem Word-Wert) macht eine Schiebe-Operation nötig:

```
SpReg:= BefCode shl 8 + Adresse;
```

Nach diesen Vorbereitungen ist die Unit »ass.pas« (Listing 10) funktionsbereit. Nun nützt Ihnen der Assembler aber herzlich wenig, wenn Sie keinen Editor zum Eingeben des Quelltextes haben. Definieren Sie also einen Editor für den Assembler. Natürlich muß er nicht so komfortabel sein wie der des Turbo-Pascal-Systems. Gemäß der Deklaration des Textpuffers im Assembler können Sie sich von vornherein auf eine bestimmte Zeilenlänge (nämlich 6 Zeichen) und eine begrenzte Zeilenanzahl (16 Zeilen) beschränken. Sieht man genau hin, dann ist der geforderte Editor von derselben Bauart wie die beiden Editoren »SpeicherEditor« und »RegisterEditor« aus den MAC-Aktionen. Schreiben Sie ihn deshalb einfach von dort ab, und Sie erhalten die Prozedur »editieren« in Listing 11 (Zeile 25 bis 62).

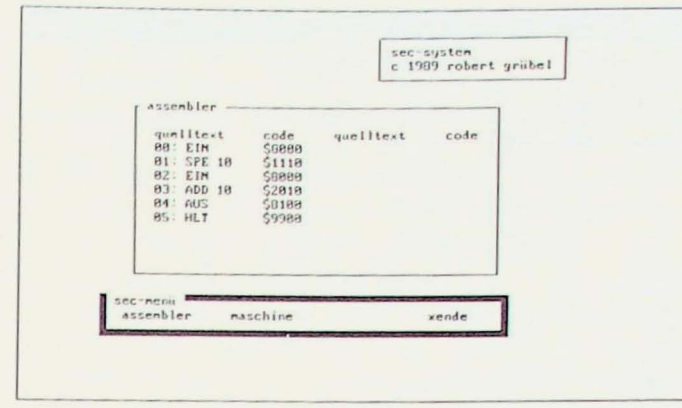


Bild 5. Bildschirmaufbau nach dem Assemblieren eines Programms

Die Definition des Assemblers und des Editors verlangt natürlich auch nach einer Oberfläche, von der aus sich die Aktionen auslösen lassen. Die Unit »assakt.pas« (Listing 11) ist das Resultat dieser Forderungen. Sie enthält zusätzlich zwei Prozeduren zum Speichern und Laden von Quellprogrammen in eine beziehungsweise aus einer Textdatei.
Weil man mit dem Assembler nur in Verbindung mit MAC sinnvoll arbeiten kann, werden die vorgestellten Aktionen in ein übergeordnetes Steuerprogramm eingebaut. Dadurch läßt sich der Assembler und die Maschine als integriertes System benutzen. Der Menüaufbau ist vom MAC-System her bekannt und muß daher nicht näher besprochen werden. Es gibt jetzt zwei Menüebenen: Die obere – als »SecMenue« bezeichnet – dient der Auswahl, ob man mit dem Assembler oder der Maschine arbeiten will. Die beiden darunterliegenden (»AssMenue« und »MacMenue«) stellen dann die jeweiligen Aktionen dieser Teile zur Verfügung (Listing 12). Bild 4 veranschaulicht die Strukturzusammenhänge des SEC-Systems, während Bild 5 den Bildschirmaufbau für eine typische Situation beim Arbeiten mit dem System wiedergibt.
Vielleicht hat das vorgestellte Programm bei Ihnen Lust und Interesse am »Computerbauen« geweckt. Für das System gibt es sicher noch viele Verwendungsbereiche. Einige Änderungs- und Erweiterungsmöglichkeiten sollen hier deshalb aufgegriffen und etwas konkretisiert werden:

- Für eine Änderung des Befehlssatzes ist die Änderung der Befehlskonstanten in der Unit »mac.pas« nötig. Diese bilden dann die neuen Case-Labels für die Verzweigungen in der Prozedur »BefehlsAusfuehrung«. Die Anpassung des Assemblers an neue Befehle geschieht durch eine Eintragung der gewählten Befehlskürzel in das Array »MacBefehle«.
- Man kann sich überlegen, vergleichend mit realen Mikroprozessoren, wo sich im vorgestellten Prozessor die ALU (arithmetisch-logische Einheit), das Steuerwerk und so weiter befinden müßten und diese Einheiten als Subeinheiten herausarbeiten.
- Zum Abstieg auf die Bitebene lassen sich die Register als Bit-Arrays definieren, um beispielsweise eine Multiplikation auf Schiebebefehle zurückzuführen. Wer sich nach dem guten alten Kernspeicher sehnt, definiert den Hauptspeicher als dreidimensionales Bit-Array und paßt die Schreib- und Lese-Operationen an.
- Zur Umgestaltung des Assemblers wird man vor allem den Quelltext zeichenweise bearbeiten, damit er nicht formatiert vorliegen muß.
- Wie wäre es mit einer Emulation? Da läuft dann plötzlich der 8080 auf dem 80386-Prozessor – aber vielleicht funktionieren ja auch vier parallel?

(Robert Grübel/zi)

Literatur: Gerhard Seegmüller, »Einführung in die Sytemprogrammierung«, Bibliographisches Institut, 1974.

Name	alle Listings
Sprache	Turbo Pascal 4.0
Grafikkarte	jede
Besonderheiten	keine

```
[DZ4]    1: { Programmname: SPE.PAS      }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[RX3]    5: {//  Speicher-Unit  //}
[ØK1]    6: unit SpE;
[SK1]    7: interface
[RNØ]    8: var
[NØ2]    9:   SpReg: word;
[ØØØ]   1Ø:
[PT5]   11: procedure HSpSchreiben(adr: byte);
[V25]   12: procedure HSpLesen(adr: byte);
[ØØØ]   13:
[4L2]   14: implementation
[ØØØ]   15:
[RNØ]   16: var
[BL2]   17:   SpAdrReg: byte;
[TX4]   18:   HSp     : array[$ØØ..$1F] of word;
[ØØØ]   19:
[PT5]   2Ø: procedure HSpSchreiben(adr: byte);
[MUØ]   21:   begin
[DB6]   22:     SpAdrReg:= adr; HSp[SpAdrReg]:= SpReg
[ØN3]   23:   end {HSpSchreiben};
[ØØØ]   24:
[V25]   25: procedure HSpLesen(adr: byte);
[MUØ]   26:   begin
[DB6]   27:     SpAdrReg:= adr; SpReg:= HSp[SpAdrReg]
[5Ø3]   28:   end {HSpLesen};
[ØØØ]   29:
[RNØ]   3Ø: var
[MN1]   31:   adr: Ø..$1F;
[MUØ]   32: begin
[P55]   33:   for adr:= Ø to $1F do HSp[adr]:= Ø
[KNØ]   34: end.
```

Listing 1. »spe.pas« stellt die Realisierung der Speicher-Unit dar

```
[4X4]    1: { Programmname: EAE.PAS      }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[C54]    5: {//  Ein/Ausgabe-Unit  //}
[UE1]    6: unit eae;
[SK1]    7: interface
[A14]    8: uses CRT, InOut, Strng;
[RNØ]    9: var
[CY1]   1Ø:   EAReg: word;
[NN2]   11: procedure eing;
[VN2]   12: procedure ausg;
[ØØØ]   13:
[4L2]   14: implementation
[ØØØ]   15:
[NN2]   16: procedure eing;
[RNØ]   17:   var
[5N1]   18:    x, y: byte;
[MUØ]   19:   begin
[MN1]   2Ø:     CursorOn;
[TF4]   21:     write(output, '   eingabe: ');
[XY3]   22:     x:= wherex; y:= wherey;
[UN5]   23:     EAReg:= Ø; ReadHexWord(EAReg, $FFFF);
[842]   24:     gotoxy(x, y);
[A25]   25:     writeln('$', WordToHex(EAReg));
[2P1]   26:     CursorOff
[ZNØ]   27:   end;
[ØØØ]   28:
[VN2]   29: procedure ausg;
[MUØ]   3Ø:   begin
[9Ø5]   31:     writeln(output, '   ausgabe: $',
[8X2]   32:             WordToHex(EAReg))
[ZNØ]   33:   end;
[ØØØ]   34:
[KNØ]   35: end.
```

Listing 2. Die Unit »eae.pas« definiert die Ein-/Ausgabe-Einheit

```
[CX4]    1: { Programmname: MAC.PAS      }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[LT2]    5: {//  mac-Unit  //}
```

```
[GF1]    6: unit Mac;
[SK1]    7: interface
[G82]    8: uses SpE, EaE;
[STØ]    9:   type
[883]   1Ø:     tRegisters= record
[V52]   11:       BefZae: word;
[912]   12:       Akku  : word;
[C72]   13:       AdrReg: word;
[S12]   14:       OpReg : word
[ZNØ]   15:     end;
[RNØ]   16:   var
[S52]   17:     BefZae: byte;
[ØØØ]   18:
[PN3]   19:   procedure prozessor;
[VX5]   2Ø:   procedure RegisterWerte(var Regs:
[N42]   21:                           tRegisters);
[ØØØ]   22:
[4L2]   23: implementation
[ØØØ]   24:
[RNØ]   25: var
[912]   26:   Akku  : word;
[972]   27:   AdrReg: byte;
[B52]   28:   OpReg : byte;
[ØØØ]   29:
[PN3]   3Ø: procedure prozessor;
[6XØ]   31:   const
[Y71]   32:     LER= $ØØ;
[4Y3]   33:     LAD= $1Ø; SPE= $11; NUM= $12;
[UV3]   34:     ADD= $2Ø; SUB= $21; NEG= $24;
[3T2]   35:     SPR= $6Ø; SPN= $61;
[NN2]   36:     EIN= $8Ø; AUS= $81;
[ØA1]   37:     HLT= $99;
[RNØ]   38:   var
[SM3]   39:     halt,  fehler: boolean;
[ØØØ]   4Ø:
[S77]   41:   procedure BefehlsAusfuehrung(OpReg: byte;
[PB2]   42:                                AdrReg: byte);
[MUØ]   43:     begin
[L22]   44:       case OpReg of
[UYØ]   45:         LER: ;
[2M1]   46:         LAD: begin
[XW2]   47:                HSpLesen(AdrReg);
[LY1]   48:                Akku:= SpReg
[ZNØ]   49:              end;
[1P1]   5Ø:         SPE: begin
[D62]   51:                SpReg:= Akku;
[EF3]   52:                HSpSchreiben(AdrReg)
[ZNØ]   53:              end;
[293]   54:         NUM: Akku:= AdrReg;
[NL1]   55:         ADD: begin
[XW2]   56:                HSpLesen(AdrReg);
[GZ2]   57:                Akku:= Akku+ SpReg
[ZNØ]   58:              end;
[4P1]   59:         SUB: begin
[XW2]   6Ø:                HSpLesen(AdrReg);
[KZ2]   61:                Akku:= Akku- SpReg
[ZNØ]   62:              end;
[PW2]   63:         NEG: Akku:= -Akku;
[AE3]   64:         SPR: BefZae:= AdrReg;
[V63]   65:         SPN: if Akku= Ø then
[AL2]   66:                BefZae:= AdrReg;
[MN1]   67:         EIN: begin
[7W2]   68:                eing; Akku:= EAReg
[ZNØ]   69:              end;
[4R1]   7Ø:         AUS: begin
[SX2]   71:                EAReg:= Akku; ausg
[ZNØ]   72:              end;
[7S2]   73:         HLT: halt:= true
[R93]   74:         else fehler:= true;
[VX1]   75:       end {case}
[ØK4]   76:     end {BefehlsAusfuehrung};
[ØØØ]   77:
[NA3]   78:   begin {prozessor}
[AH4]   79:     halt:= false; fehler:= false;
[P31]   8Ø:     repeat
[MW2]   81:       HSpLesen(BefZae);
[5X1]   82:       inc(BefZae);
[XX2]   83:       OpReg:= hi(SpReg);
[C83]   84:       AdrReg:= lo(SpReg);
[9S5]   85:       BefehlsAusfuehrung(OpReg, AdrReg);
[UC3]   86:     until halt or fehler;
[683]   87:   end {prozessor};
[ØØØ]   88:
[VX5]   89: procedure RegisterWerte(var Regs:
[N42]   9Ø:                          tRegisters);
[MUØ]   91:   begin
[3D3]   92:     Regs.BefZae:= BefZae;
[YT2]   93:     Regs.Akku:= Akku;
[BA3]   94:     Regs.OpReg:= OpReg;
[3A3]   95:     Regs.AdrReg:= AdrReg
[ØY3]   96:   end {RegisterWerte};
[ØØØ]   97:
[KNØ]   98: end.
```

Listing 3. Die Unit »mac.pas« simuliert den Prozessor des Systems

```
[LC5]     1: { Programmname: MACTEST.PAS   }
[LG4]     2: { (C) 1989 Everts & Hagedorn }
[ZB4]     3: {           DOS International }
[ØØØ]     4:
[PS2]     5: {//  mac-test  //}
[ZW2]     6: program macTest;
[ØM3]     7: uses CRT, EAE, SpE, Mac;
[ØØØ]     8:
[MUØ]     9: begin
[DB1]    1Ø:   clrscr;
[YØ5]    11:   Spreg:= $8ØØØ; HSpSchreiben($ØØ);
[KØ5]    12:   Spreg:= $111Ø; HSpSchreiben($Ø1);
[615]    13:   Spreg:= $8ØØØ; HSpSchreiben($Ø2);
[TØ5]    14:   Spreg:= $2Ø1Ø; HSpSchreiben($Ø3);
[K15]    15:   Spreg:= $81ØØ; HSpSchreiben($Ø4);
[T25]    16:   Spreg:= $99ØØ; HSpSchreiben($Ø5);
[PN3]    17:   BefZae:= Ø; prozessor;
[X11]    18:   readln
[KNØ]    19: end.
```

Listing 4. Mit »mactest.pas« wird MAC überprüft

```
[G45]     1: { Programmname: MACSYS.PAS    }
[LG4]     2: { (C) 1989 Everts & Hagedorn }
[ZB4]     3: {           DOS International }
[ØØØ]     4:
[853]     5: {//  mac-System  //}
[V83]     6: program macSystem;
[TH5]     7: uses CRT, InOut, Windows, MacAkt;
[ØØØ]     8:
[NV5]     9: {/////////// Hilfsroutinen ////////////}
[F54]    1Ø: procedure ProgrammEnde;
[MUØ]    11:   begin
[AU5]    12:     StandardWindow; clrscr; CursorOn
[DM2]    13:   end {PrgEnde};
[ØØØ]    14:
[3M3]    15: procedure CopyRight;
[MUØ]    16:   begin
[PU6]    17:     StandardWindow; SetColors(GroundColor,
[A32]    18:     WriteColor);
[Ø83]    19:     clrscr; CursorOff;
[PB4]    2Ø:     OpenWindow('', 45, 2, 22, 2, '-');
[RV3]    21:     writeln(' mac-system');
[DG4]    22:     write(' c 1989 robert grübel')
[U13]    23:   end; {CopyRight}
[ØØØ]    24:
[YH5]    25: {/////////////// mac-menü ////////////}
[4D3]    26: procedure MacMenue;
[RNØ]    27:   var
[T12]    28:     MacBef: char;
[MUØ]    29:   begin
[P31]    3Ø:     repeat
[2S3]    31:       OpenWindow('mac-menü',
[M32]    32:                  1Ø, 21, 5Ø, 1, ' ');
[VW4]    33:       write('  start    programm ein'+
[UM4]    34:             'tragen    register   xende');
[3S2]    35:       MacBef:= readkey;
[KS3]    36:       CloseWindow(5, 6, 75, 13);
[3A2]    37:       case MacBef of
[9B3]    38:         's': ProgrammStart;
[PB3]    39:         'p': SpeicherEditor;
[MB3]    4Ø:         'r': RegisterEditor
[4MØ]    41:       end
[WK2]    42:     until MacBef= 'x'
[BX2]    43:   end {MacMenue};
[ØØØ]    44:
[965]    45: {////////////////////////////////////////}
[MUØ]    46: begin
[8U1]    47:   CopyRight;
[TH1]    48:   MacMenue;
[A62]    49:   ProgrammEnde
[KNØ]    5Ø: end.
```

Listing 5. Durch »macsys.pas« erhält MAC eine Menüsteuerung

```
[C35]     1: { Programmname: INOUT.PAS     }
[LG4]     2: { (C) 1989 Everts & Hagedorn }
[ZB4]     3: {           DOS International }
[ØØØ]     4:
[X43]     5: {//  InOut-Unit  //}
[Ø22]     6: unit InOut;
[SK1]     7: interface
[PA3]     8: uses CRT, DOS, Strng;
[6XØ]     9: const
[5S4]    1Ø:   NUL= ^@;       BEL= ^G;       BS = ^H;
[NP4]    11:   LF= ^J;        FF= ^L;        CR= ^M;
```

Listing 6. Die Unit »inout.pas« stellt erweiterte Ein- und Ausgaberoutinen zur Verfügung

Geprüfte Listing-Eingabe

Alle Programmlistings werden in der DOS nach einem speziellen Verfahren mit Prüfsummen und Zeilennummern versehen und abgedruckt. Mit einem dafür vorgesehenen Prüfsummenprogramm (»SCheck«) können Sie dadurch die Korrektheit der abgetippten Listings überprüfen. Bitte beachten Sie daher die folgenden Hinweise, wenn Sie Listings aus der DOS abtippen.

Alle abgedruckten Listings sind mit Prüfsummen und Zeilennummern versehen. Jede Zeile beginnt mit einer dreistelligen Prüfsumme, die in eckige Klammern gesetzt ist. Diese Prüfsumme geben Sie beim Abtippen nicht mit ein. Auf die Prüfsumme folgt die Zeilennummer, die mit einem Doppelpunkt abgeschlossen wird. Auch die Zeilennummer geben Sie nicht mit ein. Nach dem Doppelpunkt folgt ein Leerzeichen und darauf dann der eigentliche Text der Programmzeile. Nur diesen letzten Teil der abgedruckten Zeile geben Sie beim Abtippen ein.

Bei allen Programmzeilen wird im Abdruck nach jeweils 55 Zeichen ein Zeilenumbruch vorgenommen. Die umgebrochene Teilzeile erhält dabei weder Prüfsumme noch Zeilennummer. Bei der Eingabe geben Sie umgebrochene Zeilen bitte in einem Stück ein. Drücken Sie also nicht die Return-Taste, wenn ein Zeilenumbruch erscheint, sondern nur, wenn eine neue Zeilennummer auftaucht.

Als Ausnahme werden Listings in GW-Basic ohne zusätzliche Zeilennummer abgedruckt, weil derartige Listings Zeilennummern automatisch als Bestandteil der Sprache enthalten müssen. In diesem Falle erscheint die Zeilennummer ohne Doppelpunkt und muß unbedingt mit abgetippt werden.

Um das Programm in diesem Computer zu bekommen, können Sie einen beliebigen Texteditor verwenden, der reinen ASCII-Text erzeugen kann. Das Programm ist richtig eingegeben, wenn Sie den Text beim Auflisten mittels Type-Kommando unter MS-DOS korrekt lesen können. Wie Sie die eingegebenen Programme starten, das hängt ganz vom Typ des Programms ab. Für die meisten Listings brauchen Sie einen Compiler, um den Programmtext in Maschinensprache zu übersetzen. Beachten Sie unbedingt die Hinweise in der Programmbeschreibung. Beim Abtippen von Programmen sind Tippfehler kaum zu vermeiden. Mit »SCheck«, dem Prüfsummen-Programm der DOS, können Sie die abgetippten Listings aber auf eventuelle Fehler überprüfen. Das Programm ist in Turbo Pascal 3.0 geschrieben und im Quellcode und als ausführbares Programm auf jeder Databox-Diskette der DOS International enthalten. Es erlaubt nicht nur die Diagnose von fehlerhaft eingegebenen Zeilen, sondern ermöglicht mit einem integrierten Editor auch gleich die Korrektur fehlerhafter Zeilen.

(ev)

```
[RX4]  12:   RET= CR;       ESC= ^[;       DEL= #127;
[527]  13:   LftArrow= ^S; RgtArrow= ^D; UpArrow= ^E;
[SW5]  14:   DnArrow= ^X;  PgUp= ^R;      PgDn= ^C;
[AE5]  15:   Home= ^Q;     EndKey= ^W;     INS= ^V;
[PM3]  16:   rxESC: boolean= false;
[7K3]  17:   rxSpecKey: char= NUL;
[1M5]  18:   CharFilter: set of char= [#1..#255];
[RNØ]  19: var
[S63]  2Ø:   GroundColor: byte;
[Y13]  21:   WriteColor : byte;
[PW2]  22:   SpecColor  : byte;
[PD3]  23:   Regs       : registers;
[ØØØ]  24:
[VF3]  25: procedure CursorOn;
[AK3]  26: procedure CursorOff;
[UZ4]  27: procedure SetColors(BackCol,
[RN2]  28:                     TextCol: byte);
[V14]  29: function ReadChar: char;
[ZA7]  3Ø: procedure ReadString(var rsString: string;
[EU2]  31:                      MaxLength: byte);
[1R6]  32: procedure ReadUpcString(var rsString:
[GC1]  33:                         string;
[EU2]  34:                         MaxLength: byte);
[AS6]  35: procedure ReadWord(var WordValue: word;
[ØR2]  36:                    MaxValue: word);
[ØE7]  37: procedure ReadHexWord(var WordValue: word;
[ØR2]  38:                      MaxValue: word);
[ØØØ]  39:
[4L2]  4Ø: implementation
[ØØØ]  41:
[6XØ]  42: const
[RM2]  43:   StdCursor= $Ø3Ø9;
[RM2]  44:   AltCursor= $Ø3Ø5;
[SL2]  45:   OffCursor= $26Ø7;
[ØØØ]  46:
[VF3]  47: procedure CursorOn;
[MUØ]  48:   begin
[F42]  49:     Regs.AX:= 256;
[8H3]  5Ø:     Regs.CX:= StdCursor;
[R52]  51:     intr(16, Regs)
[K13]  52:   end {CursorOn};
[ØØØ]  53:
[AK3]  54: procedure CursorOff;
[MUØ]  55:   begin
[F42]  56:     Regs.AX:= 256;
[EG3]  57:     Regs.CX:= OffCursor;
[R52]  58:     intr(16, Regs)
[Ø63]  59:   end {CursorOff};
[ØØØ]  6Ø:
[9F4]  61: function GetCursor: word;
[MUØ]  62:   begin
[M82]  63:     Regs.AX:= $3ØØ;
[3D2]  64:     intr(16, Regs);
[LH3]  65:     GetCursor:= Regs.CX;
[XF1]  66:   end {};
[ØØØ]  67:
[GK5]  68: procedure SetCursor(mode: word);
[MUØ]  69:   begin
[F42]  7Ø:     Regs.AX:= 256;
[1K2]  71:     Regs.CX:= mode;
[R52]  72:     intr(16, Regs)
[Y63]  73:   end {SetCursor};
[ØØØ]  74:
[W54]  75: procedure ChangeCursor;
[RNØ]  76:   var
[RX1]  77:     mode: word;
[MUØ]  78:   begin
[PU2]  79:     mode:= GetCursor;
[D33]  8Ø:     if mode= StdCursor
[AC4]  81:     then SetCursor(AltCursor)
[TC4]  82:     else SetCursor(StdCursor)
[EP3]  83:   end {ChangeCursor};
[ØØØ]  84:
[UZ4]  85: procedure SetColors(BackCol,
[RN2]  86:                     TextCol: byte);
[MUØ]  87:   begin
[324]  88:     textbackground(BackCol);
[873]  89:     textcolor(TextCol)
[T53]  9Ø:   end {SetColors};
[ØØØ]  91:
[V14]  92: function ReadChar: char;
[RNØ]  93:   var
[RE1]  94:     ch : char;
[MUØ]  95:   begin
[U22]  96:     ch:= readkey;
[AE1]  97:     if ch= NUL
[ZN1]  98:     then begin
[U22]  99:       ch:= readkey;
[LM1]  1ØØ:      case ch of
[A72]  1Ø1:        #71: ch:= Home;
[WT2]  1Ø2:        #72: ch:= UpArrow;
[V92]  1Ø3:        #73: ch:= PgUp;
[Ø23]  1Ø4:        #75: ch:= LftArrow;
[133]  1Ø5:        #77: ch:= RgtArrow;
[MF2]  1Ø6:        #79: ch:= EndKey;
[HR2]  1Ø7:        #8Ø: ch:= DnArrow;
[E82]  1Ø8:        #81: ch:= PgDn;
[PZ1]  1Ø9:        #82: ch:= INS;
[YS1]  11Ø:        #83: ch:= DEL
[132]  111:      else ch:= NUL
[4MØ]  112:    end
[ZNØ]  113:  end;
[VD2]  114:  ReadChar := ch;
[YW2]  115: end {ReadChar};
[ØØØ]  116:
[6Ø7]  117: procedure ReadStr(var rsString: string;
[AR2]  118:                    MaxLength: byte;
[D92]  119:                    upc: boolean);
[RNØ]  12Ø: var
[GØ2]  121:   ch        : char;
[HV2]  122:   position  : byte;
[BA2]  123:   x, y, i   : byte;
[UF3]  124:   new, insrt: boolean;
[L13]  125:   CursorMode: word;
[MUØ]  126: begin
[VV5]  127:   SetColors(GroundColor, SpecColor);
[4X5]  128:   Cursormode:= GetCursor; CursorOn;
[XY3]  129:   x:= wherex; y:= wherey;
[L94]  13Ø:   new:= true; insrt:= true;
[F55]  131:   rxESC:= false; rxSpecKey:= NUL;
[Z65]  132:   position:= length(rsString)+ 1;
[DS1]  133:   if upc then
[CH5]  134:     rsString:= UpcString(rsString);
[P31]  135:   repeat
[Y45]  136:     gotoxy(x, y); write(rsString);
[EM4]  137:     for i:= byte(rsString[Ø])+ 1
[ØD4]  138:             to MaxLength do write('_');
[A24]  139:     gotoxy(x+ position- 1, y);
[N82]  14Ø:     ch:= ReadChar;
[6B4]  141:     if upc then ch:= upcase(ch);
[LM1]  142:     case ch of
[WYØ]  143:       CR  : ;
[DX4]  144:       BS  : if position> 1 then begin
[XD2]  145:               dec(position);
[UT2]  146:               delete(rsString,
[GA2]  147:                      position- 1, 1)
[ZNØ]  148:             end;
[AX1]  149:       INS : begin
[G73]  15Ø:               insrt:= not insrt;
[T42]  151:               ChangeCursor
[ZNØ]  152:             end;
[D63]  153:       ESC : rxESC:= true;
[973]  154:       DEL : if position<=
[LW3]  155:               byte(rsString[Ø]) then
[UT2]  156:             delete(rsString,
[A22]  157:                    position, 1);
[AV4]  158:       LftArrow: if position> 1 then
[XD2]  159:                   dec(position);
[T34]  16Ø:       RgtArrow: if position<=
[LW3]  161:                   byte(rsString[Ø]) then
[2F2]  162:                 inc(position);
[375]  163:       UpArrow: rxSpecKey:= UpArrow;
[Ø45]  164:       DnArrow: rxSpecKey:= DnArrow;
[BD3]  165:       Home  : position:=  1;
[DN5]  166:       EndKey: position:= byte(rsString
[S41]  167:                              [Ø])+ 1;
[G64]  168:       PgUp  : rxSpecKey:= PgUp;
[EY3]  169:       PgDn  : rxSpecKey:= PgDn
[ZRØ]  17Ø:     else
[RV4]  171:       if ch in CharFilter then begin
[ØN2]  172:         if new then begin
[ND4]  173:           rsString:= ''; position:= 1;
[TY1]  174:           new:= false
[ZNØ]  175:         end;
[5K2]  176:         if not insrt and
[P65]  177:           (position<= byte(rsString[Ø]))
[ZN1]  178:         then begin
[DD4]  179:           rsString[position]:= ch;
[H92]  18Ø:           inc(position)
[4MØ]  181:         end
[WP1]  182:         else begin
[E95]  183:           if MaxLength> byte(rsString[Ø])
[ZN1]  184:           then begin
[VB3]  185:             insert(ch, rsString,
[KT1]  186:                    position);
[H92]  187:             inc(position)
[4MØ]  188:           end
[4MØ]  189:         end
[4MØ]  19Ø:       end;
[ZNØ]  191:     until (ch= CR) or rxESC or
[TY1]  192:           (rxSpecKey<> NUL);
[7Z3]  193:   SetColors(GroundColor, WriteColor);
[F23]  194:   gotoxy(x, y); write(rsString);
[Z56]  195:   for i:= byte(rsString[Ø])+ 1
[Y45]  196:           to MaxLength do write(' ');
[EM4]  197:   SetCursor(Cursormode)
[1B4]  198: end {ReadStr};
[TS3]  199:
[LN2]  2ØØ: procedure ReadString(var rsString: string;
[ZA7]  2Ø1:                      MaxLength: byte);
[EU2]  2Ø2: begin
[MUØ]  2Ø3:   ReadStr(rsString, MaxLength, false)
[W36]  2Ø4: end {ReadString};
[L43]  2Ø5:
[ØØØ]  2Ø6:
[1R6]  2Ø7: procedure ReadUpcString(var rsString:
[V63]  2Ø8:                         string; MaxLength:
[711]  2Ø9:                         byte);
```

```
[MUØ]  211:   begin
[ZY5]  212:     ReadStr(rsString, MaxLength, true)
[L43]  213:   end {ReadString};
[ØØØ]  214:
[6P6]  215: procedure ReadWrd(var WordValue: word;
[DR2]  216:                    MaxValue: word;
[6A2]  217:                    hex: boolean);
[RNØ]  218:   var
[S34]  219:     rwString, HStr : string;
[G63]  22Ø:     MaxLength      : byte;
[RC2]  221:     ch             : char;
[263]  222:     position       : byte;
[WM2]  223:     x, y, i        : byte;
[274]  224:     new, insrt, byt: boolean;
[MC3]  225:     CursorMode     : word;
[G64]  226:     SaveCharF      : set of char;
[763]  227:     fpos           : integer;
[MUØ]  228:   begin
[VV5]  229:     SetColors(GroundColor, SpecColor);
[4X5]  23Ø:     Cursormode:= GetCursor; CursorOn;
[XY3]  231:     x:= wherex; y:= wherey;
[XM2]  232:     if hex then begin
[EM2]  233:       write('$'); inc(x)
[ZNØ]  234:     end;
[LV5]  235:     new:= true; insrt:= true; fpos:= Ø;
[F55]  236:     rxESC:= false; rxSpecKey:= NUL;
[LØ4]  237:     SaveCharF:= CharFilter;
[XM2]  238:     if hex then begin
[5B3]  239:       byt:= maxValue<= 255;
[2Ø4]  24Ø:       if byt then MaxLength:= 2
[N63]  241:       else MaxLength:= 4;
[ZT1]  242:       if byt then
[6G5]  243:         rwString:= ByteToHex(WordValue)
[BM6]  244:       else rwString:= WordToHex(WordValue);
[KG4]  245:       CharFilter:= ['Ø'..'9', 'A'..'F',
[551]  246:                     'a'..'f']
[4MØ]  247:     end
[WP1]  248:     else begin
[8Y5]  249:       MaxLength:= trunc(ln(abs(MaxValue))/
[WD1]  25Ø:                        ln(1Ø))+ 1;
[DC4]  251:       str(WordValue, rwString);
[LL3]  252:       CharFilter:= ['Ø'..'9']
[ZNØ]  253:     end;
[375]  254:     position:= length(rwString)+ 1;
[P31]  255:     repeat
[A55]  256:       gotoxy(x, y); write(rwString);
[YM4]  257:       for i:= byte(rwString[Ø])+ 1
[ØD4]  258:               to MaxLength do write('_');
[A24]  259:       gotoxy(x+ position- 1, y);
[N82]  26Ø:       ch:= ReadChar;
[5F3]  261:       if ch in ['a'..'f'] then
[TH2]  262:         ch:= upcase(ch);
[LM1]  263:       case ch of
[WYØ]  264:         CR  : ;
[DX4]  265:         BS  : if position> 1 then begin
[XD2]  266:                 dec(position);
[ØS4]  267:                 delete(rwString,position,1)
[ZNØ]  268:               end;
[AX1]  269:         INS : begin
[G73]  27Ø:                 insrt:= not insrt;
[T42]  271:                 ChangeCursor
[ZNØ]  272:               end;
[D63]  273:         ESC : rxESC:= true;
[973]  274:         DEL : if position<=
[3X3]  275:                   byte(rwString[Ø]) then
[2U2]  276:                 delete(rwString,
[A22]  277:                        position, 1);
[AV4]  278:         LftArrow: if position> 1 then
[XD2]  279:                     dec(position);
[T34]  28Ø:         RgtArrow: if position<=
[3X3]  281:                       byte(rwString[Ø]) then
[2F2]  282:                     inc(position);
[375]  283:         UpArrow: rxSpecKey:= UpArrow;
[Ø45]  284:         DnArrow: rxSpecKey:= DnArrow;
[BD3]  285:         Home  : position:=  1;
[TN5]  286:         EndKey: position:= byte(rwString
[S41]  287:                                [Ø])+ 1;
[G64]  288:         PgUp  : rxSpecKey:= PgUp;
[EY3]  289:         PgDn  : rxSpecKey:= PgDn
[ZRØ]  29Ø:       else
[S43]  291:         if ch in CharFilter
[ZN1]  292:         then begin
[ØN2]  293:           if new then begin
[1E4]  294:             rwString:= ''; position:= 1;
[TY1]  295:             new:= false
[ZNØ]  296:           end;
[5K2]  297:           if not insrt and
[675]  298:              (position<= byte(rwString[Ø]))
[ZN1]  299:           then begin
[TD4]  3ØØ:             rwString[position]:= ch;
[H92]  3Ø1:             inc(position)
[4MØ]  3Ø2:           end
[WP1]  3Ø3:           else begin
[P95]  3Ø4:             if MaxLength> byte(rwString[Ø])
[ZN1]  3Ø5:             then begin
[C25]  3Ø6:               insert(ch,rwString,position);
[H92]  3Ø7:               inc(position)
[4MØ]  3Ø8:             end
[4MØ]  3Ø9:           end
[4MØ]  31Ø:         end
[ZNØ]  311:       end;
[MØ2]  312:       new:= false;
[2E5]  313:       if rwString= '' then WordValue:= Ø
[LS2]  314:       else if hex then
[UL5]  315:         Val('$'+ rwString, WordValue, fpos)
[EØ6]  316:        else Val(rwString, WordValue, fpos);
[ZE5]  317:       if WordValue> MaxValue then begin
[XM2]  318:         if hex then begin
[ZT1]  319:           if byt then
[WC4]  32Ø:             HStr:= ByteToHex(MaxValue)
[UF5]  321:           else HStr:= WordToHex(MaxValue)
[4MØ]  322:         end
[Ø64]  323:         else str(MaxValue, HStr);
[EF5]  324:         fpos:= difString(rwString, HStr)
[ZNØ]  325:       end;
[G73]  326:       if fpos> Ø then begin
[P35]  327:         position:= fpos; insrt:= false
[4MØ]  328:       end
[8C4]  329:     until ((ch= CR) or rxESC or
[E15]  33Ø:           (rxSpecKey<> NUL)) and (fpos= Ø);
[Z56]  331:     SetColors(GroundColor, WriteColor);
[XM2]  332:     if hex then begin
[PØ4]  333:       gotoxy(x- 1, y); write('$')
[ZNØ]  334:     end;
[842]  335:     gotoxy(x, y);
[XM2]  336:     if hex then begin
[ZT1]  337:       if byt then
[ØU4]  338:         write(ByteToHex(WordValue))
[ØH5]  339:       else write(WordToHex(WordValue))
[4MØ]  34Ø:     end
[SH5]  341:     else write(WordValue:MaxLength);
[DZ3]  342:     CharFilter:= SaveCharF;
[TS3]  343:     SetCursor(Cursormode)
[9N2]  344:   end {ReadWrd};
[ØØØ]  345:
[AS6]  346: procedure ReadWord(var WordValue: word;
[ØR2]  347:                    MaxValue: word);
[MUØ]  348:   begin
[2Ø6]  349:     ReadWrd(WordValue, MaxValue, false)
[BZ2]  35Ø:   end {ReadWord};
[ØØØ]  351:
[ØE7]  352: procedure ReadHexWord(var WordValue: word;
[ØR2]  353:                       MaxValue: word);
[MUØ]  354:   begin
[5V5]  355:     ReadWrd(WordValue, MaxValue, true)
[BZ2]  356:   end {ReadWord};
[ØØØ]  357:
[U64]  358: begin {initialisation}
[YD4]  359:   if mem[$4Ø:$49]= 7 then begin
[TD2]  36Ø:   {monochrom}
[ZD3]  361:     GroundColor:= black;
[K54]  362:     WriteColor:= lightgray;
[DY2]  363:     SpecColor:= white
[4MØ]  364:   end
[WP1]  365:   else begin
[FP1]  366:   {color}
[UH3]  367:     GroundColor:= white;
[UB3]  368:     WriteColor:= black;
[8P2]  369:     SpecColor:= red
[4MØ]  37Ø:   end
[KNØ]  371: end.
```

Listing 6. Die Unit »inout.pas« stellt erweiterte Ein- und Ausgaberoutinen zur Verfügung

```
[RF5]    1: { Programmname: WINDOWS.PAS  }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[ØN3]    5: {//  Windows-Unit  //}
[XA2]    6: unit Windows;
[SK1]    7: interface
[5H1]    8: uses CRT;
[ØØØ]    9:
[5B6]   1Ø: procedure OpenWindow(Titel: string;
[2Z1]   11:                      XPos, YPos,
[HØ3]   12:                      Cols, Lines: byte;
[3Z2]   13:                      FrameChars: char);
[PR5]   14: procedure CloseWindow(XPos, YPos,
[763]   15:                       Cols, Lines: byte);
[YE4]   16: procedure StandardWindow;
[ØØØ]   17:
[4L2]   18: implementation
[ØØØ]   19:
[5B6]   2Ø: procedure OpenWindow(Titel: string;
[2Z1]   21:                      XPos, YPos,
[HØ3]   22:                      Cols, Lines: byte;
[3Z2]   23:                      FrameChars: char);
[RNØ]   24:   var
```

Listing 7. Die Unit »windows.pas« definiert einfache Fenster-Funktionen

```
[ST3]   25:     lx,  ly,  rx,  ry   : byte;
[B34]   26:     owa, uwa, lse, rse  : char;
[Y14]   27:     loe, roe, lue, rue  : char;
[AT2]   28:     i                   : byte;
[MUØ]   29:   begin
[254]   3Ø:     lx:= XPos- 1; ly:= YPos- 1;
[7N5]   31:     rx:= XPos+ Cols; ry:= YPos+ Lines;
[643]   32:     case FrameChars of
[8C1]   33:       '-': begin
[2B3]   34:              owa:= #196;  uwa:= #196;
[PA3]   35:              lse:= #179;  rse:= #179;
[K83]   36:              loe:= #218;  roe:= #191;
[Ø83]   37:              lue:= #192;  rue:= #217
[ZNØ]   38:            end;
[SD1]   39:       '=': begin
[H93]   4Ø:              owa:= #2Ø5;  uwa:= #2Ø5;
[6A3]   41:              lse:= #186;  rse:= #186;
[G83]   42:              loe:= #2Ø1;  roe:= #187;
[Ø83]   43:              lue:= #2ØØ;  rue:= #188
[4MØ]   44:            end
[WP1]   45:       else begin
[193]   46:              owa:= #223;  uwa:= #22Ø;
[V93]   47:              lse:= #219;  rse:= #219;
[Z83]   48:              loe:= #219;  roe:= #219;
[E83]   49:              lue:= #219;  rue:= #219
[4MØ]   5Ø:            end
[ZNØ]   51:     end;
[ZU2]   52:     StandardWindow;
[8C4]   53:     gotoxy(lx, ly); write(loe);
[UR3]   54:     for i:= lx+ 1 to rx- 1 do
[UW1]   55:       write(owa);
[6X1]   56:     write(roe);
[ØS4]   57:     if length(Titel)> Ø then begin
[7R2]   58:       gotoxy(lx+ 1, ly);
[6X2]   59:       write(' '+ Titel+ ' ')
[ZNØ]   6Ø:     end;
[FU4]   61:     for i:= ly+ 1 to ry- 1 do begin
[M34]   62:       gotoxy(lx, i); write(lse);
[VY3]   63:       gotoxy(rx, i); write(rse)
[5X1]   64:     end {for};
[MC4]   65:     gotoxy(lx, ry); write(lue);
[UR3]   66:     for i:= lx+ 1 to rx- 1 do
[KX1]   67:       write(uwa);
[CX1]   68:     write(rue);
[EC3]   69:     if Lines> Ø then begin
[FØ5]   7Ø:       window(lx+ 1, ly+ 1, rx- 1, ry- 1);
[141]   71:       clrscr
[4MØ]   72:     end
[373]   73:   end {OpenWindow};
[ØØØ]   74:
[PR5]   75: procedure CloseWindow(XPos, YPos,
[763]   76:                        Cols, Lines: byte);
[MUØ]   77:   begin
[4T3]   78:     window(XPos- 1, YPos- 1,
[R24]   79:            XPos+ Cols, YPos+ Lines);
[ØØ4]   8Ø:     clrscr; StandardWindow
[2D3]   81:   end {CloseWindow};
[ØØØ]   82:
[YE4]   83: procedure StandardWindow;
[MUØ]   84:   begin
[8P2]   85:     window(1, 1, 8Ø, 25)
[5X3]   86:   end {StandardWindow};
[ØØØ]   87:
[KNØ]   88: end.
```

Listing 7. Die Unit »windows.pas« definiert einfache Fenster-Funktionen

```
[X35]    1: { Programmname: STRNG.PAS    }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[153]    5: {//  Strng-Unit  //}
[422]    6: unit Strng;
[SK1]    7: interface
[ØØØ]    8:
[DK7]    9: function UpcString(UpcStr: string): string;
[YN8]   1Ø: function DifString(DifStr1, DifStr2: string
            ): byte;
[V87]   11: function ByteToHex(ByteVal: byte): string;
[897]   12: function WordToHex(WordVal: word): string;
[ØØØ]   13:
[4L2]   14: implementation
[ØØØ]   15:
[DK7]   16: function UpcString(UpcStr: string): string;
[RNØ]   17:   var
[Z71]   18:     i: byte;
[MUØ]   19:   begin
[YX4]   2Ø:     for i:= 1 to length(UpcStr) do
[5Z2]   21:       case UpcStr[i] of
[UV3]   22:         'ö': UpcStr[i]:= 'Ö';
[1W3]   23:         'ü': UpcStr[i]:= 'Ü';
[DT3]   24:         'ä': UpcStr[i]:= 'Ä';
[4X1]   25:         'ß': begin
```

```
[6R2]   26:                UpcStr[i]:= 'S';
[LE3]   27:                insert('S', UpcStr, i)
[4MØ]   28:              end
[E66]   29:            else UpcStr[i]:= Upcase(UpcStr[i])
[ZNØ]   3Ø:       end;
[G73]   31:     UpcString:= UpcStr
[F73]   32:   end {UpcString};
[ØØØ]   33:
[A46]   34: function DifString(DifStr1, DifStr2:
[VB2]   35:                    String): byte;
[RNØ]   36:   var
[XL1]   37:     l, i: byte;
[MUØ]   38:   begin
[DW5]   39:     DifString:= ord(DifStr1<> DifStr2);
[VE5]   4Ø:     DifString:= ord(length(DifStr1)<>
[BP2]   41:                    length(DifStr2));
[WY3]   42:     if (DifStr1<> DifStr2) and
[8M5]   43:        (length(DifStr1)= length(DifStr2))
[ZN1]   44:     then begin
[X54]   45:       l:= length(DifStr1); i:= 1;
[GA2]   46:       while (i<= l) and
[BZ4]   47:             (DifStr1[i]= DifStr2[i]) do
[951]   48:         inc(i);
[Ø62]   49:       DifString:= i
[4MØ]   5Ø:     end
[353]   51:   end {DifString};
[ØØØ]   52:
[6XØ]   53: const
[RX3]   54:   HexString: array[Ø..15]
[8X3]   55:              of char= 'Ø123456789ABCDEF';
[ØØØ]   56:
[V87]   57: function ByteToHex(ByteVal: byte): string;
[MUØ]   58:   begin
[NE6]   59:     ByteToHex:= HexString[ByteVal shr 4]+
[7E4]   6Ø:                HexString[ByteVal and $ØF]
[373]   61:   end {ByteToHex};
[ØØØ]   62:
[897]   63: function WordToHex(WordVal: word): string;
[MUØ]   64:   begin
[S36]   65:     WordToHex:= ByteToHex(hi(WordVal))+
[FU3]   66:                 ByteToHex(lo(WordVal))
[P73]   67:   end {WordToHex};
[KNØ]   68: end.
```

Listing 8. Die Unit »strng.pas« bietet Routinen zur Stringbearbeitung

```
[535]    1: { Programmname: MACAKT.PAS    }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {          DOS International }
[ØØØ]    4:
[4F4]    5: {//  Maschinen-Aktionen  //}
[832]    6: unit MacAkt;
[SK1]    7: interface
[XØ7]    8: uses CRT, InOut, Windows, Strng, SpE, Mac;
[ØØØ]    9:
[DD4]   1Ø: procedure ProgrammStart;
[UD4]   11: procedure SpeicherEditor;
[VF4]   12: procedure RegisterEditor;
[ØØØ]   13:
[4L2]   14: implementation
[ØØØ]   15:
[DD4]   16: procedure ProgrammStart;
[MUØ]   17:   begin
[FØ4]   18:     CloseWindow(1Ø, 21, 5Ø, 1);
[F82]   19:     gotoxy(15, 21);
[T8G]   2Ø:     write('[redacted]'+
[Z88]   21:           '[redacted]');
[7H4]   22:     OpenWindow('programmlauf',
[F22]   23:                25, 7, 24, 11, '-');
[5Y1]   24:     gotoxy(1, 2);
[5G3]   25:     BefZae:= Ø; prozessor
[9Z3]   26:   end {ProgrammStart};
[ØØØ]   27:
[UD4]   28: procedure SpeicherEditor;
[RNØ]   29:   var
[7D2]   3Ø:     ZeilenNr: Ø..31;
[NC2]   31:     Spalte  : byte;
[MUØ]   32:   begin
[FØ4]   33:     CloseWindow(1Ø, 21, 5Ø, 1);
[PX4]   34:     OpenWindow('beenden mit ESC',
[K32]   35:                1Ø, 21, 5Ø, Ø, ' ');
[MZ4]   36:     OpenWindow('speicher-editor',
[L22]   37:                15, 7, 47, 1Ø, '-');
[Ø35]   38:     for ZeilenNr:= Ø to $1F do begin
[MS4]   39:       Spalte:= ZeilenNr div 8* 11+ 3;
[DD3]   4Ø:       HSpLesen(ZeilenNr);
[WD5]   41:       gotoxy(Spalte, ZeilenNr mod 8+ 2);
[FM5]   42:       write(ByteToHex(ZeilenNr):3, ': $',
[FZ2]   43:             WordToHex(SpReg))
[ZNØ]   44:     end;
[222]   45:     ZeilenNr:= Ø;
[P31]   46:     repeat
[MS4]   47:       Spalte:= ZeilenNr div 8* 11+ 3;
```

```
[DD3]    48:        HSpLesen(ZeilenNr);
[WD5]    49:        gotoxy(Spalte, ZeilenNr mod 8+ 2);
[RP5]    5Ø:        write(ByteToHex(ZeilenNr):3, ': ');
[N94]    51:        ReadHexWord(SpReg, $FFFF);
[M34]    52:        HSpSchreiben(ZeilenNr);
[TV2]    53:        case rxSpecKey of
[RR4]    54:          UpArrow: if ZeilenNr= Ø then
[H12]    55:                     ZeilenNr:= 31
[N83]    56:                   else dec(ZeilenNr);
[383]    57:          PgUp   : ZeilenNr:= Ø;
[XA3]    58:          PgDn   : ZeilenNr:= 31;
[N24]    59:          else if ZeilenNr< 31 then
[L72]    6Ø:                 inc(ZeilenNr)
[XU2]    61:               else ZeilenNr:= Ø
[4MØ]    62:        end
[F42]    63:      until rxESC;
[R52]    64:      rxESC:= false
[MV3]    65:    end {SpeicherEditor};
[ØØØ]    66:
[VF4]    67: procedure RegisterEditor;
[RNØ]    68:    var
[PX2]    69:      Regs: tRegisters;
[KD2]    7Ø:      ZeilenNr: 3..9;
[MUØ]    71:    begin
[FØ4]    72:      CloseWindow(1Ø, 21, 5Ø, 1);
[PX4]    73:      OpenWindow('beenden mit ESC',
[K32]    74:                 1Ø, 21, 5Ø, Ø, ' ');
[XØ5]    75:      OpenWindow('register-editor',
[722]    76:                 15, 7, 4Ø, 11, '-');
[EF3]    77:      RegisterWerte(Regs);
[BY1]    78:      gotoxy(3, 3);
[RK4]    79:      write('Befehlszähler     :   $',
[X14]    8Ø:            ByteToHex(Regs.BefZae));
[LY1]    81:      gotoxy(3, 5);
[FB4]    82:      write('Akkumulator       :   $',
[DU3]    83:            WordToHex(Regs.Akku));
[VY1]    84:      gotoxy(3, 7);
[XM4]    85:      write('Adressregister    :   $',
[624]    86:            ByteToHex(Regs.AdrReg));
[3Z1]    87:      gotoxy(3, 9);
[B45]    88:      write('Operationsregister:   $',
[BØ4]    89:            ByteToHex(Regs.OpReg));
[522]    9Ø:      ZeilenNr:= 3;
[P31]    91:      repeat
[PD3]    92:        gotoxy(25, ZeilenNr);
[SR2]    93:        case ZeilenNr of
[U55]    94:          3: ReadHexWord(Regs.BefZae, $FF);
[E45]    95:          5: ReadHexWord(Regs.Akku, $FFFF);
[L75]    96:          7: ReadHexWord(Regs.AdrReg, $FF);
[XØ5]    97:          9: ReadHexWord(Regs.OpReg, $FF);
[ZNØ]    98:        end;
[TV2]    99:        case rxSpecKey of
[3S4]   1ØØ:          UpArrow: if ZeilenNr= 3 then
[NY1]   1Ø1:                     ZeilenNr:= 9
[XP2]   1Ø2:                   else ZeilenNr:=
[EX1]   1Ø3:                        ZeilenNr- 2;
[683]   1Ø4:          PgUp   : ZeilenNr:= 3;
[663]   1Ø5:          PgDn   : ZeilenNr:= 9;
[1Ø4]   1Ø6:          else if ZeilenNr< 9 then
[ZM3]   1Ø7:                 ZeilenNr:= ZeilenNr+ 2
[3V2]   1Ø8:               else ZeilenNr:= 3
[4MØ]   1Ø9:        end
[F42]   11Ø:      until rxESC;
[R52]   111:      rxESC:= false
[7X3]   112:    end {RegisterEditor};
[ØØØ]   113:
[KNØ]   114: end.
```

Listing 9. »macakt.pas« vereinigt alle Menü-Steueraktionen in einer Unit

```
[AØ5]     1: { Programmname: ASS.PAS       }
[LG4]     2: { (C) 1989 Everts & Hagedorn }
[ZB4]     3: {           DOS International }
[ØØØ]     4:
[WT3]     5: {//  Assembler-Unit  //}
[4H1]     6: unit Ass;
[SK1]     7: interface
[F96]     8: uses CRT, InOut, Windows, Strng, SpE;
[STØ]     9: type
[ØX6]    1Ø:   tTextPuffer= array[Ø..15] of string[6];
[ØØØ]    11:
[VB6]    12: procedure Assembler(var TextPuffer:
[4C2]    13:                     tTextPuffer);
[ØØØ]    14:
[4L2]    15: implementation
[VB6]    16: procedure Assembler(var TextPuffer:
[4C2]    17:                     tTextPuffer);
[6XØ]    18:   const
[854]    19:     MacBefehle: array[$Ø..$9F]
[N42]    2Ø:                 of string[3]=
[MB4]    21:     {typisierte Konstante}
[8S3]    22:       ('LER', '', '', '', '', '', '', '',
[S33]    23:        '', '', '', '', '', '', '', '',
[PT3]    24:        'LAD', 'SPE', 'NUM', '', '', '',
[BG3]    25:        '', '', '', '', '', '', '', '',
[144]    26:        '', 'ADD', 'SUB', '', '', 'NEG', '',
[BG3]    27:        '', '', '', '', '', '', '', '', '',
[BG3]    28:        '', '', '', '', '', '', '', '', '',
[BG3]    29:        '', '', '', '', '', '', '', '', '',
[BG3]    3Ø:        '', '', '', '', '', '', '', '', '',
[BG3]    31:        '', '', '', '', '', '', '', '', '',
[BG3]    32:        '', '', '', '', '', '', '', '', '',
[6V3]    33:        '', '', '', '', 'SPR', 'SPN', '', '',
[BG3]    34:        '', '', '', '', '', '', '', '', '',
[BG3]    35:        '', '', '', '', '', '', '', '', '',
[BG3]    36:        '', '', '', '', '', '', '', '', '',
[BR3]    37:        '', '', 'EIN', 'AUS', '', '', '', '',
[BG3]    38:        '', '', '', '', '', '', '', '', '',
[BG3]    39:        '', '', '', '', '', '', '', '', '',
[LR3]    4Ø:        '', '', 'HLT', '', '', '', '', '',
[2DØ]    41:        '');
[RNØ]    42:    var
[4F2]    43:      ZeilenNr : Ø..15;
[BH3]    44:      BefString: string[3];
[KK3]    45:      AdrString: string[2];
[DN2]    46:      BefCode  : $Ø..$99;
[VK2]    47:      Adresse  : byte;
[AD2]    48:      Spalte   : byte;
[HS2]    49:      ende     : boolean;
[NV2]    5Ø:      fcode    : integer;
[MUØ]    51:    begin
[9Ø4]    52:      OpenWindow('assembler',
[E22]    53:                 14, 7, 45, 11, '-');
[HB5]    54:      gotoxy(3, 2); write('quelltext');
[A74]    55:      gotoxy(17, 2); write('code');
[LC5]    56:      gotoxy(26, 2); write('quelltext');
[Z64]    57:      gotoxy(4Ø, 2); write('code');
[3A4]    58:      ZeilenNr:= Ø; ende:= false;
[7H5]    59:      while (ZeilenNr<= 15) and not ende do
[MUØ]    6Ø:      begin
[5Z4]    61:        BefString:= ''; AdrString:= '';
[5S3]    62:        BefCode:= Ø; Adresse:= Ø;
[RX4]    63:        BefString:= copy(TextPuffer
[NT2]    64:                         [ZeilenNr], 1, 3);
[PD3]    65:        if length(TextPuffer
[K73]    66:                  [ZeilenNr])> 4 then
[2Z4]    67:          AdrString:= copy(TextPuffer
[WT2]    68:                           [ZeilenNr], 5, 2);
[7W3]    69:        while (BefString<> '') and
[YG5]    7Ø:            (BefString<> MacBefehle[BefCode])
[X73]    71:            and (BefCode<= $99) do
[V32]    72:          inc(BefCode);
[YL5]    73:        val('$'+ AdrString, Adresse, fcode);
[545]    74:        SpReg:= BefCode shl 8+ Adresse;
[M34]    75:        HSpSchreiben(ZeilenNr);
[GR4]    76:        if ZeilenNr< 8 then Spalte:= 3
[9R2]    77:        else Spalte:= 26;
[XD5]    78:        gotoxy(Spalte, ZeilenNr mod 8+ 3);
[645]    79:        write(ByteToHex(ZeilenNr), ': ',
[P34]    8Ø:              TextPuffer[ZeilenNr]);
[8S5]    81:        gotoxy(Spalte+ 12, ZeilenNr mod 8+3);
[SU4]    82:        write('  $', WordToHex(SpReg));
[4D2]    83:        inc(ZeilenNr);
[YM3]    84:        ende:= BefString= 'HLT'
[4MØ]    85:      end
[G33]    86:    end {Assembler};
[KNØ]    87: end.
```

Listing 10. Die Assembler-Unit des Systems: »ass.pas«

```
[365]     1: { Programmname: ASSAKT.PAS    }
[LG4]     2: { (C) 1989 Everts & Hagedorn }
[ZB4]     3: {           DOS International }
[ØØØ]     4:
[9G4]     5: {//  Assembler-Aktionen  //}
[W42]     6: unit AssAkt;
[SK1]     7: interface
[L86]     8: uses CRT, InOut, Windows, Strng, Ass;
[1ZØ]     9: {$V-}
[P34]    1Ø: procedure assemblieren;
[GH3]    11: procedure editieren;
[8Y3]    12: procedure PrgTxtLaden;
[FP4]    13: procedure PrgTxtSpeichern;
[ØØØ]    14:
[4L2]    15: implementation
[RNØ]    16: var
[FT2]    17:   AssDatei  : text;
[GE4]    18:   TextPuffer: tTextPuffer;
[ØØØ]    19:
[P34]    2Ø: procedure assemblieren;
[MUØ]    21:   begin
[9R3]    22:     Assembler(TextPuffer)
[EN3]    23:   end {assemblieren};
[ØØØ]    24:
[GH3]    25: procedure editieren;
[RNØ]    26:   var
[HD2]    27:     ZeilenNr: Ø..15;
[NC2]    28:     Spalte  : byte;
[MUØ]    29:   begin
[FØ4]    3Ø:     CloseWindow(1Ø, 21, 5Ø, 1);
[PX4]    31:     OpenWindow('beenden mit ESC',
[K32]    32:                1Ø, 21, 5Ø, Ø, ' ');
[X34]    33:     OpenWindow('ass-editor',
```

Listing 11. Die Unit »assakt.pas« definiert alle Aktionen zur Nutzung des Assemblers

```
[U12]   34:                21, 7, 3Ø, 1Ø, '-');
[PZ4]   35:      for ZeilenNr:= Ø to 15 do begin
[DR4]   36:        if ZeilenNr< 8 then Spalte:= 2
[7R2]   37:        else Spalte:= 18;
[WD5]   38:        gotoxy(Spalte, ZeilenNr mod 8+ 2);
[MF5]   39:        write(ByteToHex(ZeilenNr):3, ': ',
[P34]   4Ø:              TextPuffer[ZeilenNr]);
[ZNØ]   41:      end;
[222]   42:      ZeilenNr:= Ø;
[P31]   43:      repeat
[DR4]   44:        if ZeilenNr< 8 then Spalte:= 2
[7R2]   45:        else Spalte:= 18;
[WD5]   46:        gotoxy(Spalte, ZeilenNr mod 8+ 2);
[RP5]   47:        write(ByteToHex(ZeilenNr):3, ': ');
[4B4]   48:        readupcstring(TextPuffer
[CL2]   49:                      [ZeilenNr], 6);
[TV2]   5Ø:        case rxSpecKey of
[RR4]   51:          UpArrow: if ZeilenNr= Ø then
[R12]   52:                     ZeilenNr:= 15
[N83]   53:                   else dec(ZeilenNr);
[3B3]   54:          PgUp   : ZeilenNr:= Ø;
[3B3]   55:          PgDn   : ZeilenNr:= 15;
[H24]   56:          else if ZeilenNr< 15 then
[L72]   57:                 inc(ZeilenNr)
[XU2]   58:               else ZeilenNr:= Ø
[4MØ]   59:        end
[F42]   6Ø:      until rxESC;
[R52]   61:      rxESC:= false
[E23]   62:    end {editieren};
[ØØØ]   63:
[8Y3]   64: procedure PrgTxtLaden;
[RNØ]   65:    var
[K84]   66:      AssDateiName: string[3Ø];
[HR2]   67:      ZeilenNr    : Ø..15;
[MUØ]   68:    begin
[FØ4]   69:      CloseWindow(1Ø, 21, 5Ø, 1);
[LH5]   7Ø:      OpenWindow('programmtext laden',
[S52]   71:                 15, 15, 45, 3, '-');
[FW2]   72:      AssDateiName:= '';
[NG5]   73:      gotoxy(3, 2); write('Dateiname: ');
[R65]   74:      ReadUpcString(AssDateiName, 3Ø);
[FU5]   75:      AssDateiName:= AssDateiName+ '.ASS';
[ØB5]   76:      assign(AssDatei, AssDateiName);
[E15]   77:      {$I-}; reset(AssDatei); {$I+ }
[YE2]   78:      if ioresult<> Ø
[ZN1]   79:      then begin
[MA3]   8Ø:        clrscr; gotoxy(3, 2);
[ML4]   81:        write('Datei ', AssDateiName,
[6Ø3]   82:              ' nicht gefunden.')
[4MØ]   83:      end
[WP1]   84:      else begin
[8X3]   85:        for ZeilenNr:= Ø to 15 do
[4K4]   86:          TextPuffer[ZeilenNr]:= '';
[222]   87:        ZeilenNr:= Ø;
[815]   88:        while not eof(AssDatei) do begin
[NP2]   89:          readln(AssDatei,
[P34]   9Ø:                 TextPuffer[ZeilenNr]);
[L72]   91:          inc(ZeilenNr)
[ZNØ]   92:        end;
[5H2]   93:        close(AssDatei)
[4MØ]   94:      end
[ØE3]   95:    end {PrgTxtLaden};
[ØØØ]   96:
[FP4]   97: procedure PrgTxtSpeichern;
[RNØ]   98:    var
[K84]   99:      AssDateiName: string[3Ø];
[HR2]  1ØØ:      ZeilenNr    : Ø..15;
[D43]  1Ø1:      fertig      : boolean;
[MUØ]  1Ø2:    begin
[FØ4]  1Ø3:      CloseWindow(1Ø, 21, 5Ø, 1);
[ZA6]  1Ø4:      OpenWindow('programmtext speichern',
[S52]  1Ø5:                 15, 15, 45, 3, '-');
[FW2]  1Ø6:      AssDateiName:= '';
[NG5]  1Ø7:      gotoxy(3, 2); write('Dateiname: ');
[R65]  1Ø8:      ReadupcString(AssDateiName, 3Ø);
[FU5]  1Ø9:      AssDateiName:= AssDateiName+ '.ASS';
[ØB5]  11Ø:      assign(AssDatei, AssDateiName);
[KA5]  111:      {$I-}; rewrite(AssDatei); {$I+ }
[X54]  112:      if ioresult<> Ø then begin
[MA3]  113:        clrscr; gotoxy(3, 2);
[BM4]  114:        write('die Datei kann nicht '+
[BV2]  115:              'erzeugt werden')
[4MØ]  116:      end
[WP1]  117:      else begin
[222]  118:        ZeilenNr:= Ø;
[P31]  119:        repeat
[813]  12Ø:          writeln(AssDatei,
[P34]  121:                  TextPuffer[ZeilenNr]);
[XD5]  122:          fertig:= TextPuffer[ZeilenNr]=
[9XØ]  123:                   'HLT';
[L72]  124:          inc(ZeilenNr)
[M55]  125:        until (ZeilenNr> 15) or fertig;
[5H2]  126:        close(AssDatei)
[ZNØ]  127:      end;
[274]  128:    end {PrgTxtSpeichern};
[ØØØ]  129:
[RNØ]  13Ø: var
[HD2]  131:    ZeilenNr: Ø..15;
[MUØ]  132: begin
[8X3]  133:    for ZeilenNr:= Ø to 15 do
[4K4]  134:      TextPuffer[ZeilenNr]:= '';
[KNØ]  135: end.
```

Listing 11. Die Unit »assakt.pas« definiert alle Aktionen zur Nutzung des Assemblers

```
[855]    1: { Programmname: SECSYS.PAS    }
[LG4]    2: { (C) 1989 Everts & Hagedorn }
[ZB4]    3: {           DOS International }
[ØØØ]    4:
[A63]    5: {//  sec-System  //}
[L93]    6: program secSystem;
[HT6]    7: uses CRT, InOut, Windows, MacAkt, AssAkt;
[ØØØ]    8:
[G26]    9: {////////////// hilfsroutinen //////////////}
[153]   1Ø: procedure PrgEnde;
[MUØ]   11:   begin
[AU5]   12:     StandardWindow; clrscr; CursorOn
[DM2]   13:   end {PrgEnde};
[ØØØ]   14:
[3M3]   15: procedure CopyRight;
[MUØ]   16:   begin
[ZU2]   17:     StandardWindow;
[Z56]   18:     SetColors(GroundColor, WriteColor);
[Ø83]   19:     clrscr; CursorOff;
[PB4]   2Ø:     OpenWindow('', 45, 2, 22, 2, '-');
[LW3]   21:     writeln(' sec-system');
[DG4]   22:     write(' c 1989 robert grübel')
[U13]   23:   end; {CopyRight}
[ØØØ]   24:
[2M5]   25: {////////////////////// menüs //////////////}
[4D3]   26: procedure MacMenue;
[RNØ]   27:   var
[T12]   28:     MacBef: char;
[MUØ]   29:   begin
[2S3]   3Ø:     OpenWindow('mac-menü',
[M32]   31:                1Ø, 21, 5Ø, 1, ' ');
[N35]   32:     write('  start      programm ein'+
[HL3]   33:           'tragen      register');
[3S2]   34:     MacBef:= readkey;
[3A2]   35:     case MacBef of
[9B3]   36:       's': ProgrammStart;
[PB3]   37:       'p': SpeicherEditor;
[MB3]   38:       'r': RegisterEditor
[4MØ]   39:     end
[BX2]   4Ø:   end {MacMenue};
[ØØØ]   41:
[6D3]   42: procedure AssMenue;
[RNØ]   43:   var
[R42]   44:     AssBef: char;
[MUØ]   45:   begin
[ØV3]   46:     OpenWindow('ass-menü',
[M32]   47:                1Ø, 21, 5Ø, 1, ' ');
[KR5]   48:     write('  assemblieren   editieren   '+
[N63]   49:           'laden   speichern');
[1V2]   5Ø:     AssBef:= readkey;
[RB2]   51:     case AssBef of
[9W2]   52:       'a': assemblieren;
[4D2]   53:       'e': editieren;
[TT2]   54:       'l': PrgTxtLaden;
[UF3]   55:       's': PrgTxtSpeichern
[4MØ]   56:     end
[ZY2]   57:   end {AssMenue};
[ØØØ]   58:
[ØE3]   59: procedure SecMenue;
[RNØ]   6Ø:   var
[H22]   61:     SecBef: char;
[MUØ]   62:   begin
[P31]   63:     repeat
[US3]   64:       OpenWindow('sec-menü',
[M32]   65:                  1Ø, 21, 5Ø, 1, ' ');
[WØ5]   66:       write('  assembler     maschine '+
[AK2]   67:             '                xende');
[P31]   68:       repeat
[5N2]   69:         SecBef:= readkey
[SW3]   7Ø:       until SecBef in ['a', 'm',
[A12]   71:                        'i', 'x', 'X'];
[KS3]   72:       CloseWindow(5, 6, 75, 13);
[5B2]   73:       case SecBef of
[U52]   74:         'a': AssMenue;
[TZ1]   75:         'm': MacMenue
[4MØ]   76:       end
[K44]   77:     until SecBef in ['x', 'X']
[DY2]   78:   end {SecMenue};
[ØØØ]   79:
[965]   8Ø: {////////////////////////////////////////}
[MUØ]   81: begin
[8U1]   82:   CopyRight;
[HK1]   83:   SecMenue;
[LB1]   84:   PrgEnde
[KNØ]   85: end.
```

Listing 12. Mit »secsys.pas« wird ein integriertes System aus MAC und zugehörigem Assembler realisiert

Tastatur und Treiber ohne Mystik

Nicht immer ist es mit den ASCII-Werten getan, wenn es darum geht, in eigenen Programmen Funktionen mit bestimmten Tasten auszulösen. Das Programm «Tastmon« hilft Ihnen weiter: Es macht Ihre Tastatur »transparenter«.

Tastmon liefert Ihnen auf Tastendruck schnell und vollständig genau die Information, nach der Sie sonst unangemessen lange in Handbüchern oder Zeitschriften blättern. Nach dem Start baut das Programm einen eigenen Bildschirm auf und wartet auf Tastatureingaben. Der Tastatur-Interrupt-Vektor wird auf eine eigene Routine umgeleitet, welche die Tastatureingabe direkt liest und diese – falls es sich um einen Taste-gedrückt-Code handelt – als Tastaturcode ablegt; außerdem wird eine Marke für das Hauptprogramm gesetzt, um anzuzeigen, daß der Tastaturpuffer auszulesen und die Bildschirmausgabe zu aktualisieren ist. Abschließend wird das ursprüngliche Programm zur Tastaturunterbrechung aufgerufen, das seinerseits das Treiberprogramm aktiviert. Eine Ausnahme bilden die Eingaben »Taste freigeben« sowie die Kombination [Ctrl-Alt-Del].

Erstere werden nicht als Tastaturcode abgelegt, und es erfolgt daher eine unmittelbare Verzweigung zum ursprünglichen Unterbrechungsprogramm. Die Tastenkombination [Ctrl-Alt-Del] führt zum Setzen einer Marke »Programm beenden« für das Hauptprogramm und zum Zurücksetzen beziehungsweise Freigeben der Tastatur.

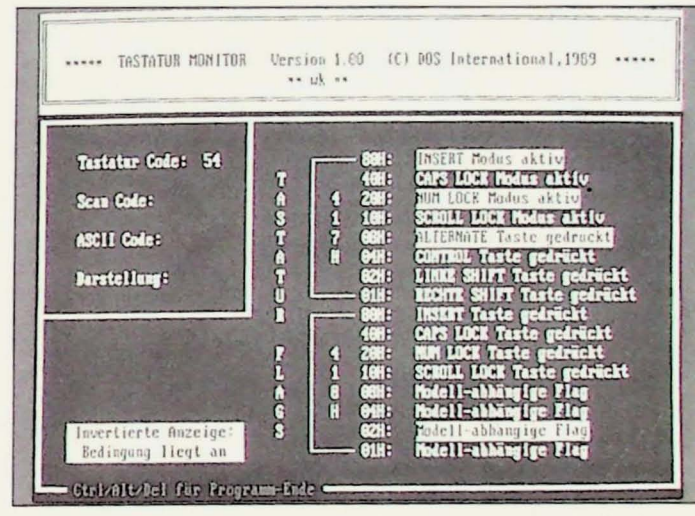
Das Hauptprogramm wartet in einer Schleife auf die vom Unterbrechungsprogramm gesetzten Marken. Die Marke »Programm beenden« bewirkt das Zurücksetzen des umgeleiteten Tastatur-Interrupt-Vektors, Löschen des Bildschirms (Mögen Sie Programme, die den Bildschirm in einem Zustand hinterlassen, der an einen Hühnerhof erinnert?) und Rückkehr zum MS-DOS-Prompt. Mit dem MS-DOS-Prompt, das heißt auf der Betriebssystemebene, gewinnt die Kombination [Ctrl-Alt-Del] wieder ihre gewohnte Bedeutung.

Seine eigentliche Existenzberechtigung (und Nützlichkeit) gewinnt das Programm Tastmon allerdings erst dann, wenn eine Tastatureingabe erfolgt und die Marke »Taste gedrückt« gesetzt ist. Führen Sie sich nochmals kurz vor Augen, in welchem Zustand Sie sich befinden und welche Informationen Ihnen für die Auswertung zur Verfügung stehen:

- Die Tastatur hat eine Unterbrechung erzwungen und die »Kennung« der – die Unterbrechung auslösenden – Taste wurde als Tastaturcode abgelegt;
- die ursprüngliche Tastatur-Unterbrechungsroutine und der Tastaturtreiber (Übersetzungsprogramm) wurden durchlaufen und haben ihre spezifischen Ergebnisse in »bekannten« Speicherzellen des Arbeitsspeichers abgelegt;
- der Tastaturzustand wurde anhand der Tastatureingaben aktualisiert (Tastaturflags, Tastaturpuffer).

Wenn Sie das Ganze bis hierher gedanklich nachvollzogen haben, werden Sie feststellen, daß Sie nur noch »die Lumpen einsammeln« müssen, um Ihre Hausaufgabe zu erledigen. Entledigen Sie sich daher der selbstgestellten Aufgabe mit dem nötigen Respekt vor dem oft geplagten Anwender und vor sich selbst: Aktion transparente Tastatur.

(Wolfgang Kühn/zi)



»Tastmon« liefert Informationen ohne langwieriges Nachschlagen in Handbüchern

```
[PV4]    1: ; Programmname: TASTMON.ASM
[WY3]    2: ; (C) 1989 Everts & Hagedorn
[8U3]    3: ;           DOS International
[ØØØ]    4:
[LW5]    5: ;=====================================
[RX1]    6: ; Autor: wk
[P3Ø]    7: ;
[6N5]    8: ; Monitor-Programm zur Anzeige von
[GS4]    9: ; Tastatur-Codes und -Status
[LW5]   1Ø: ;=====================================
[ØØØ]   11:
[395]   12: code       segment para public 'code'
[WP1]   13: org        1ØØh
[638]   14: assume cs:code, ds:code, ss:nothing, es:not
            hing
[ØØØ]   15:
[LW5]   16: ;=====================================
[7Z2]   17: start:     jmp     init
[LW5]   18: ;=====================================
[ØØØ]   19:
[6X3]   2Ø: ; Daten  Declarationen:
[6Y3]   21:         ; Orig.Vektor Tastatur-
[C84]   22:         ; Unterbrechungsroutine
[TA2]   23: intØ9      dw    Ø,Ø
[724]   24:         ; Adr.Bildschirmspeicher
[9B2]   25: kbb_sbsg   dw    Ø
[HZ3]   26:         ; Tastatur Status Flags
[C22]   27: kbb_417    db    Ø
[FY3]   28:         ; Erw.Tast.Status Flags
[G22]   29: kbb_418    db    Ø
[MF3]   3Ø:         ; Programm-Ende Marke
[BZ1]   31: quit       db    Ø
[PK2]   32:         ; Display Marke
[G92]   33: dspflag    db    Ø
[ØØØ]   34:
[3C4]   35: ;-------------------------------------
[2E3]   36: ; Bildschirm Layout:
[ØØØ]   37:
[M7F]   38: l_ØØ    db  '╔══════════════════════════════════════╗'
[Y7E]   39:         db  '╚══════════════════════════════════════╝'
[2YD]   4Ø:         db  '╔═══════════════════════════════════╗'
[GW3]   41: l_Ø1    db  '╚                                   ╝'
[AS2]   42:         db  '                                     '
[B93]   43:         db  '                                    ╗'
[BM5]   44: l_Ø2    db  '╚  *****  TASTATUR MONITOR '
[674]   45:         db  '  Version 1.ØØ   (C) DOS In'
[4U4]   46:         db  'ternational,1989  *****  ╔'
[KW3]   47: l_Ø3    db  '╚                                   ╝'
[X23]   48:         db  '     ** wk **                        '
[B93]   49:         db  '                                    ╗'
[P7F]   5Ø: l_Ø4    db  '╚══════════════════════════════════════╝'
[Y7E]   51:         db  '╔══════════════════════════════════════╗'
[6YD]   52:         db  '╚═══════════════════════════════════╝'
[EKØ]   53:         db  Ø
[H7F]   54: l_Ø5    db  '╔═══════════════════════════════════════'
[Y7E]   55:         db  '╚═══════════════════════════════════════'
```

Listing. Tastatur und Tastaturtreiber – mit »tastmon.asm« enträtselt

Name	tastmon.asm
Sprache	MASM ab Version 4.0, TASM
Grafikkarte	jede
Besonderheiten	keine

```
[2YD]   56:           db  '┌──────────────────────────┐'
[XF4]   57: l_Ø6      db  '│                          │'
[AS2]   58:           db  '│                          │'
[B93]   59:           db  '│                          │'
[ZU5]   6Ø: l_Ø7      db  '│    Tastatur Code:        │'
[D46]   61:           db  '│          ┌──── 8ØH:   INSERT'
[9F4]   62:           db  '│ Modus aktiv              │'
[ZF4]   63: l_Ø8      db  '│                          │'
[YV3]   64:           db  '│   T   │      4ØH:   CAPS L'
[6P4]   65:           db  'OCK Modus aktiv             │'
[WB5]   66: l_Ø9      db  '│    Scan Code:            │'
[YX3]   67:           db  '│   A   │  4   2ØH:   NUM LO'
[WN4]   68:           db  'CK Modus aktiv              │'
[VF4]   69: l_1Ø      db  '│                          │'
[M34]   7Ø:           db  '│   S   │  1   1ØH:   SCROLL'
[6S4]   71:           db  ' LOCK Modus aktiv           │'
[PD5]   72: l_11      db  '│    ASCII Code:           │'
[C24]   73:           db  '│   T   │  7   Ø8H:   ALTERN'
[335]   74:           db  'ATE Taste gedrückt          │'
[XF4]   75: l_12      db  '│                          │'
[W24]   76:           db  '│   A   │  H   Ø4H:   CONTRO'
[HY4]   77:           db  'L Taste gedrückt            │'
[CR5]   78: l_13      db  '│    Darstellung:          │'
[PZ3]   79:           db  '│   T   │      Ø2H:   LINKE '
[RC5]   8Ø:           db  'SHIFT Taste gedrückt        │'
[ZF4]   81: l_14      db  '│                          │'
[U56]   82:           db  '│   U   └──── Ø1H:   RECHTE'
[4L5]   83:           db  ' SHIFT Taste gedrückt       │'
[Z5F]   84: l_15      db  '├──────────────────────────┘'
[H76]   85:           db  '│   R   ┌──── 8ØH:   INSERT'
[EV4]   86:           db  ' Taste gedrückt             │'
[TW3]   87: l_16      db  '│                          │'
[PS3]   88:           db  '│       │      4ØH:   CAPS L'
[F25]   89:           db  'OCK Taste gedrückt          │'
[UW3]   9Ø: l_17      db  '│                          │'
[8Y3]   91:           db  '│   F   │  4   2ØH:   NUM LO'
[H75]   92:           db  'CK Taste gedrückt           │'
[VW3]   93: l_18      db  '│                          │'
[634]   94:           db  '│   L   │  1   1ØH:   SCROLL'
[L75]   95:           db  ' LOCK Taste gedrückt        │'
[WW3]   96: l_19      db  '│                          │'
[Ø14]   97:           db  '│   A   │  8   Ø8H:   Modell'
[7U4]   98:           db  '-abhängige Flag             │'
[RW3]   99: l_2Ø      db  '│                          │'
[M14]   1ØØ:          db  '│   G   │  H   Ø4H:   Modell'
[7U4]   1Ø1:          db  '-abhängige Flag             │'
[YX5]   1Ø2: l_21     db  '│    Invertierte Anzeige:  │'
[Ø14]   1Ø3:          db  '│   S   │      Ø2H:   Modell'
[7U4]   1Ø4:          db  '-abhängige Flag             │'
[ØH5]   1Ø5: l_22     db  '│     Bedingung liegt an   │'
[616]   1Ø6:          db  '│       └──── Ø1H:   Modell'
[7U4]   1Ø7:          db  '-abhängige Flag             │'
[UW3]   1Ø8: l_23     db  '│                          │'
[AS2]   1Ø9:          db  '│                          │'
[B93]   11Ø:          db  '│                          │'
[Y7F]   111: l_24     db  '└──────────────────────────┘'
[Y7E]   112:          db  '┌──────────────────────────┐'
[6YD]   113:          db  '└──────────────────────────┘'
[EKØ]   114:          db  Ø
[3H5]   115: l_211    db  ' Invertierte Anzeige: ',Ø
[545]   116: l_221    db  '  Bedingung liegt an  ',Ø
[UC5]   117: l_Ø71    db  '│    Tastatur Code:      ',Ø
[RX4]   118: l_Ø91    db  '│    Scan Code:          ',Ø
[HZ4]   119: l_111    db  '│    ASCII Code:         ',Ø
[795]   12Ø: l_131    db  '│    Darstellung:        ',Ø
[VZ5]   121: l_241    db  ' Ctrl/Alt/Del für Programm'
[BS1]   122:          db  '-Ende ',Ø
[9Y4]   123: l_Ø7r    db  'INSERT Modus aktiv',Ø
[8F5]   124: l_Ø8r    db  'CAPS LOCK Modus aktiv',Ø
[NB5]   125: l_Ø9r    db  'NUM LOCK Modus aktiv',Ø
[8T5]   126: l_1Ør    db  'SCROLL LOCK Modus aktiv',Ø
[X76]   127: l_11r    db  'ALTERNATE Taste gedrückt',Ø
[RØ6]   128: l_12r    db  'CONTROL Taste gedrückt',Ø
[ØM6]   129: l_13r    db  'LINKE SHIFT Taste gedrückt',Ø
[RD6]   13Ø: l_14r    db  'RECHTE SHIFT Taste gedrückt',Ø
[5P5]   131: l_15r    db  'INSERT Taste gedrückt',Ø
[C56]   132: l_16r    db  'CAPS LOCK Taste gedrückt',Ø
[XW5]   133: l_17r    db  'NUM LOCK Taste gedrückt',Ø
[YH6]   134: l_18r    db  'SCROLL LOCK Taste gedrückt',Ø
[4L5]   135: l_19r    db  'Modell-abhängige Flag',Ø
[ZK5]   136: l_2Ør    db  'Modell-abhängige Flag',Ø
[ØL5]   137: l_21r    db  'Modell-abhängige Flag',Ø
[1L5]   138: l_22r    db  'Modell-abhängige Flag',Ø
[LW5]   139: ;=====================================
[T86]   14Ø: ;   Tastatur-Unterbrechungs-Routine:
[SØ3]   141: myintØ9:  push   ds
[W91]   142:           push   cs
[R71]   143:           pop    ds
[4A1]   144:           push   ax
[5A1]   145:           push   bx
[6A1]   146:           push   cx
[7A1]   147:           push   dx
[991]   148:           push   di
[S91]   149:           push   si
[Y91]   15Ø:           push   es
[613]   151:         ; Tastatur-Eingabe
[S71]   152:         ; lesen
[HH1]   153:           in     al,6Øh
[4P1]   154:           mov    ah,al
[HN1]   155:           and    ah,8Øh
[P33]   156:         ; Taste gedrückt ?
[2P1]   157:           cmp    ah,ØØh
[ZHØ]   158:         ; ja
[FP1]   159:           je     key_p
[EP2]   16Ø:           mov    dspflag,Ø1h
[NP3]   161:         ; nein: Tastenfreigabe
[CD2]   162:           jmp    myintØ9p
[AR3]   163:         ; DEL-Taste gedrückt ?
[H93]   164: key_p:    cmp    al,53h
[H12]   165:           je     key_del
[CA2]   166:         ; nein: weiter
[W12]   167:           jmp    key_ok
[2K2]   168:         ; ja: prüfen ob
[3W2]   169:         ; Ctrl/Alt anliegt
[Z43]   17Ø: key_del:  push   ax
[W42]   171:           mov    ax,ØØ4Øh
[AR1]   172:           mov    es,ax
[8A4]   173:           mov    al,byte ptr es:[ØØ17h]
[CN1]   174:           and    al,Øch
[XP1]   175:           cmp    al,Øch
[YC2]   176:           je     set_quit
[Y71]   177:           pop    ax
[W12]   178:           jmp    key_ok
[FP3]   179:         ; Ctrl/Alt/Del liegt an:
[353]   18Ø: set_quit: pop    ax
[WN4]   181:         ;  Programm-Ende Marke setzen
[4C2]   182:           mov    quit,Ø1h
[AB4]   183:         ;  Tastatur zurücksetzen
[KH1]   184:           in     al,61h
[4P1]   185:           mov    ah,al
[HK1]   186:           or     al,8Øh
[6T1]   187:           out    61h,al
[8P1]   188:           mov    al,ah
[6T1]   189:           out    61h,al
[4S1]   19Ø:           mov    al,2Øh
[ØT1]   191:           out    2Øh,al
[CE2]   192:           jmp    myintØ9x
[ZN4]   193:         ; Taste (nicht Ctrl/Alt/Del)
[U62]   194:         ; gedrückt:
[NM3]   195: key_ok:   call   zcb_1ØØ
[ØA4]   196:           mov    byte ptr l_Ø71+21,ah
[HA4]   197:           mov    byte ptr l_Ø71+22,al
[ØC5]   198:         ; gelesenen Tastencode anzeigen
[UU2]   199:         ; restl. Anzeigen
[R93]   2ØØ:         ; im kleinen Fenster
[3T1]   2Ø1:         ; löschen -
[4S1]   2Ø2:           mov    al,2Øh
[KA4]   2Ø3:           mov    byte ptr l_Ø91+21,al
[LA4]   2Ø4:           mov    byte ptr l_Ø91+22,al
[EA4]   2Ø5:           mov    byte ptr l_111+21,al
[DA4]   2Ø6:         ; werden im Hauptprogramm
[SN2]   2Ø7:         ; neu generiert.
[FA4]   2Ø8:           mov    byte ptr l_111+22,al
[HA4]   2Ø9:           mov    byte ptr l_131+22,al
[EP2]   21Ø:           mov    dspflag,Ø1h
[2M2]   211: myintØ9p: pushf
[6M4]   212:         ; Tastatur-Treiber bedienen
[EMØ]   213:           cli
[BB4]   214:           call   cs:dword ptr[intØ9]
[WE2]   215:         ; Register für
[FB4]   216:         ; unterbrochenes Programm
[S43]   217:         ; wiederherstellen
[9Z2]   218: myintØ9x: pop    es
[K71]   219:           pop    si
[371]   22Ø:           pop    di
[181]   221:           pop    dx
[Ø81]   222:           pop    cx
[Z71]   223:           pop    bx
[Y71]   224:           pop    ax
[Ø52]   225:         ; Ende eigenes
[LC4]   226:          ; Unterbrechungs-Programm
[R71]   227:           pop    ds
[GRØ]   228:           iret
[3C4]   229: ;-------------------------------------
[DM5]   23Ø: ; Hauptprogramm: Darstellung von
[PN3]   231: ;                Tastatur-Code
[XB3]   232: ;                und -Status
[ØØØ]   233:
[S65]   234:         ; Programm-Ende Marke gesetzt ?
[VF3]   235: dloop:    cmp    quit,Ø1h
[GPØ]   236:         ;  ja:
[BH1]   237:           je     term
[GY3]   238:         ;  nein: Anzeige-Zyklus
[KX1]   239:           jmp    dloop1
```

```
[MM2]  24Ø:          ; Programm-Ende:
[AS1]  241: term:      cli
[UM4]  242:          ; Original-Vektor Tastatur-
[494]  243:          ; Unterbrechungs-Routine
[S43]  244:          ; wiederherstellen
[PR1]  245:            xor     ax,ax
[AR1]  246:            mov     es,ax
[292]  247:            mov     ax,int09
[SH4]  248:   .        mov     word ptr es:[ØØ24h],ax
[HD2]  249:            mov     ax,int09+2
[UH4]  25Ø:            mov     word ptr es:[ØØ26h],ax
[6PØ]  251:            sti
[NN3]  252:          ; Bildschirm löschen
[Y32]  253:            call    zst_1ØØ
[RS1]  254:            mov     ah,4ch
[ZY2]  255:          ; zurück zu DOS
[SB1]  256:            int     21h
[884]  257:          ; Display Marke gesetzt ?
[YØ4]  258: dloop1:    cmp     dspflag,ØØh
[5E2]  259:          ;  nein: warten
[UP1]  26Ø:            je      dloop1
[564]  261:          ;  ja: Marke zurücksetzen
[CP2]  262:            mov     dspflag,ØØh
[P3Ø]  263:          ;
[W42]  264:            mov     ax,ØØ4Øh
[8W3]  265:          ; Tastatur-Statusflags
[471]  266:          ; holen
[AR1]  267:            mov     es,ax
[D24]  268:            mov     ax,word ptr es:ØØ17h
[FF2]  269:            mov     kbb_418,ah
[MF2]  27Ø:            mov     kbb_417,al
[9F5]  271:          ; Flags-Anzeige aufbereiten und
[WT4]  272:          ; in Video-Speicher schreiben
[9S1]  273:            mov     bl,3Øh
[MT1]  274:            test    al,8Øh
[PV1]  275:            jnz     l_Ø7rd
[LS1]  276:            mov     bl,1bh
[4A4]  277: l_Ø7rd:    mov     ax,kbb_sbsg
[452]  278:            mov     dx,Ø73Øh
[G72]  279:            lea     si,l_Ø7r
[L32]  28Ø:            call    zos_1ØØ
[9S1]  281:            mov     bl,3Øh
[UF2]  282:            mov     al,kbb_417
[4T1]  283:            test    al,4Øh
[UV1]  284:            jnz     l_Ø8rd
[LS1]  285:            mov     bl,1bh
[5A4]  286: l_Ø8rd:    mov     ax,kbb_sbsg
[552]  287:            mov     dx,Ø83Øh
[L72]  288:            lea     si,l_Ø8r
[L32]  289:            call    zos_1ØØ
[9S1]  29Ø:            mov     bl,3Øh
[UF2]  291:            mov     al,kbb_417
[WS1]  292:            test    al,2Øh
[YV1]  293:            jnz     l_Ø9rd
[LS1]  294:            mov     bl,1bh
[6A4]  295: l_Ø9rd:    mov     ax,kbb_sbsg
[652]  296:            mov     dx,Ø93Øh
[P72]  297:            lea     si,l_Ø9r
[L32]  298:            call    zos_1ØØ
[9S1]  299:            mov     bl,3Øh
[UF2]  3ØØ:            mov     al,kbb_417
[SS1]  3Ø1:            test    al,1Øh
[XU1]  3Ø2:            jnz     l_1Ørd
[LS1]  3Ø3:            mov     bl,1bh
[1A4]  3Ø4: l_1Ørd:    mov     ax,kbb_sbsg
[E52]  3Ø5:            mov     dx,Øa3Øh
[X62]  3Ø6:            lea     si,l_1Ør
[L32]  3Ø7:            call    zos_1ØØ
[9S1]  3Ø8:            mov     bl,3Øh
[UF2]  3Ø9:            mov     al,kbb_417
[WS1]  31Ø:            test    al,Ø8h
[1V1]  311:            jnz     l_11rd
[LS1]  312:            mov     bl,1bh
[2A4]  313: l_11rd:    mov     ax,kbb_sbsg
[F52]  314:            mov     dx,Øb3Øh
[Ø72]  315:            lea     si,l_11r
[L32]  316:            call    zos_1ØØ
[9S1]  317:            mov     bl,3Øh
[UF2]  318:            mov     al,kbb_417
[SS1]  319:            test    al,Ø4h
[5V1]  32Ø:            jnz     l_12rd
[LS1]  321:            mov     bl,1bh
[3A4]  322: l_12rd:    mov     ax,kbb_sbsg
[G52]  323:            mov     dx,Øc3Øh
[372]  324:            lea     si,l_12r
[L32]  325:            call    zos_1ØØ
[9S1]  326:            mov     bl,3Øh
[UF2]  327:            mov     al,kbb_417
[PS1]  328:            test    al,Ø2h
[9V1]  329:            jnz     l_13rd
[LS1]  33Ø:            mov     bl,1bh
[4A4]  331: l_13rd:    mov     ax,kbb_sbsg
[H52]  332:            mov     dx,Ød3Øh
[672]  333:            lea     si,l_13r
[L32]  334:            call    zos_1ØØ
[9S1]  335:            mov     bl,3Øh
[UF2]  336:            mov     al,kbb_417
[NS1]  337:            test    al,Ø1h
[DV1]  338:            jnz     l_14rd
[LS1]  339:            mov     bl,1bh
[5A4]  34Ø: l_14rd:    mov     ax,kbb_sbsg
[K52]  341:            mov     dx,Øe3Øh
[972]  342:            lea     si,l_14r
[L32]  343:            call    zos_1ØØ
[9S1]  344:            mov     bl,3Øh
[WF2]  345:            mov     al,kbb_418
[MT1]  346:            test    al,8Øh
[HV1]  347:            jnz     l_15rd
[LS1]  348:            mov     bl,1bh
[6A4]  349: l_15rd:    mov     ax,kbb_sbsg
[L52]  35Ø:            mov     dx,Øf3Øh
[C72]  351:            lea     si,l_15r
[L32]  352:            call    zos_1ØØ
[9S1]  353:            mov     bl,3Øh
[WF2]  354:            mov     al,kbb_418
[4T1]  355:            test    al,4Øh
[NV1]  356:            jnz     l_16rd
[LS1]  357:            mov     bl,1bh
[7A4]  358: l_16rd:    mov     ax,kbb_sbsg
[152]  359:            mov     dx,1Ø3Øh
[F72]  36Ø:            lea     si,l_16r
[L32]  361:            call    zos_1ØØ
[9S1]  362:            mov     bl,3Øh
[WF2]  363:            mov     al,kbb_418
[WS1]  364:            test    al,2Øh
[TV1]  365:            jnz     l_17rd
[LS1]  366:            mov     bl,1bh
[8A4]  367: l_17rd:    mov     ax,kbb_sbsg
[252]  368:            mov     dx,113Øh
[K72]  369:            lea     si,l_17r
[L32]  37Ø:            call    zos_1ØØ
[9S1]  371:            mov     bl,3Øh
[WF2]  372:            mov     al,kbb_418
[SS1]  373:            test    al,1Øh
[XV1]  374:            jnz     l_18rd
[LS1]  375:            mov     bl,1bh
[9A4]  376: l_18rd:    mov     ax,kbb_sbsg
[352]  377:            mov     dx,123Øh
[N72]  378:            lea     si,l_18r
[L32]  379:            call    zos_1ØØ
[9S1]  38Ø:            mov     bl,3Øh
[WF2]  381:            mov     al,kbb_418
[WS1]  382:            test    al,Ø8h
[1W1]  383:            jnz     l_19rd
[LS1]  384:            mov     bl,1bh
[AA4]  385: l_19rd:    mov     ax,kbb_sbsg
[452]  386:            mov     dx,133Øh
[S72]  387:            lea     si,l_19r
[L32]  388:            call    zos_1ØØ
[9S1]  389:            mov     bl,3Øh
[WF2]  39Ø:            mov     al,kbb_418
[SS1]  391:            test    al,Ø4h
[ØV1]  392:            jnz     l_2Ørd
[LS1]  393:            mov     bl,1bh
[5A4]  394: l_2Ørd:    mov     ax,kbb_sbsg
[552]  395:            mov     dx,143Øh
[Z62]  396:            lea     si,l_2Ør
[L32]  397:            call    zos_1ØØ
[9S1]  398:            mov     bl,3Øh
[WF2]  399:            mov     al,kbb_418
[PS1]  4ØØ:            test    al,Ø2h
[4V1]  4Ø1:            jnz     l_21rd
[LS1]  4Ø2:            mov     bl,1bh
[6A4]  4Ø3: l_21rd:    mov     ax,kbb_sbsg
[652]  4Ø4:            mov     dx,153Øh
[272]  4Ø5:            lea     si,l_21r
[L32]  4Ø6:            call    zos_1ØØ
[9S1]  4Ø7:            mov     bl,3Øh
[WF2]  4Ø8:            mov     al,kbb_418
[NS1]  4Ø9:            test    al,Ø1h
[8V1]  41Ø:            jnz     l_22rd
[LS1]  411:            mov     bl,1bh
[7A4]  412: l_22rd:    mov     ax,kbb_sbsg
[752]  413:            mov     dx,163Øh
[572]  414:            lea     si,l_22r
[L32]  415:            call    zos_1ØØ
[LS1]  416:            mov     bl,1bh
[CX2]  417:            mov     ax,kbb_sbsg
[Y42]  418:            mov     dx,Ø7ØØh
[S62]  419:            lea     si,l_Ø71
[L32]  42Ø:            call    zos_1ØØ
[ØØØ]  421:
[293]  422:          ; Tastatur-Eingabe?
[NR1]  423:            mov     ah,Ø1h
[1C1]  424:            int     16h
[FYØ]  425:          ; nein
[GB2]  426:            jz      kbb_supd
[ZHØ]  427:          ; ja
[AF2]  428:            jmp     kbb_char
[MZ3]  429:          ; generiere Ausgaben im
```

Listing. Tastatur und Tastaturtreiber – mit »tastmon.asm« enträtselt

```
[RU2]  430:         ; kleinen Fenster
[TM3]  431: kbb_supd: mov    bl,1bh
[CX2]  432:           mov    ax,kbb_sbsg
[952]  433:           mov    dx,Øb00h
[862]  434:           lea    si,l_111
[L32]  435:           call   zos_1ØØ
[LS1]  436:           mov    bl,1bh
[CX2]  437:           mov    ax,kbb_sbsg
[B52]  438:           mov    dx,Ød00h
[E62]  439:           lea    si,l_131
[L32]  44Ø:           call   zos_1ØØ
[LS1]  441:           mov    bl,1bh
[CX2]  442:           mov    ax,kbb_sbsg
[Ø52]  443:           mov    dx,Ø9ØØh
[Y62]  444:           lea    si,l_Ø91
[L32]  445:           call   zos_1ØØ
[GU1]  446:           jmp    dloop
[DM4]  447:         ; Tastatur-Eingabe holen und
[X24]  448:         ; für Anzeige aufbereiten
[TE3]  449: kbb_char: mov    ah,ØØh
[1C1]  45Ø:           int    16h
[4A1]  451:           push   ax
[812]  452:           call   zcb_1ØØ
[Y94]  453:           mov    byte ptr l_111+21,ah
[FA4]  454:           mov    byte ptr l_111+22,al
[Y71]  455:           pop    ax
[4A1]  456:           push   ax
[AP1]  457:           cmp    al,ØØh
[DX1]  458:           jne    kbb_dc
[4S1]  459:           mov    al,2Øh
[LC5]  46Ø: kbb_dc:   mov    byte ptr l_131+22,al
[Y71]  461:           pop    ax
[5N1]  462:           xchg   ah,al
[812]  463:           call   zcb_1ØØ
[2A4]  464:           mov    byte ptr l_Ø91+21,ah
[LA4]  465:           mov    byte ptr l_Ø91+22,al
[9G2]  466:           jmp    kbb_supd
[ØØØ]  467:
[3C4]  468: ;-------------------------------------
[VN4]  469: ; Konvertiere Byte nach ASCII
[VW2]  47Ø: ; Eing:  (AL)  =  Byte
[EA4]  471: ; Ausg:  (AX)  =  Byte in ASCII
[ØØØ]  472:
[KØ3]  473: zcb_1ØØ   proc   near
[5A1]  474:           push   bx
[4A1]  475:           push   ax
[312]  476:           call   zba_1ØØ
[HR1]  477:           mov    bx,ax
[Y71]  478:           pop    ax
[7H1]  479:           shr    al,1
[7H1]  48Ø:           shr    al,1
[7H1]  481:           shr    al,1
[7H1]  482:           shr    al,1
[312]  483:           call   zba_1ØØ
[5N1]  484:           xchg   ah,al
[GP1]  485:           mov    al,bl
[Z71]  486:           pop    bx
[3PØ]  487:           ret
[G12]  488: zcb_1ØØ   endp
[ØØØ]  489:
[3C4]  49Ø: ;-------------------------------------
[555]  491: ; Konvertiere Nibble nach ASCII
[WH4]  492: ; Eing: (AL) bits Ø - 3 = nibble
[7R4]  493: ; Ausg: (AL) = ASCII Zeichen Ø-F
[ØØØ]  494:
[BØ3]  495: zba_1ØØ   proc   near
[Z12]  496:           jmp    zba_1Ø2
[7K4]  497: zba_1Ø1   db     'Ø123456789ABCDEF'
[AW2]  498: zba_1Ø2:  push   bx
[FN1]  499:           and    al,Øfh
[ØE2]  5ØØ:           lea    bx,zba_1Ø1
[CRØ]  5Ø1:           xlat
[Z71]  5Ø2:           pop    bx
[3PØ]  5Ø3:           ret
[912]  5Ø4: zba_1ØØ   endp
[ØØØ]  5Ø5:
[3C4]  5Ø6: ;-------------------------------------
[YV5]  5Ø7: ; Ausgabe ASCII String auf Bilschirm
[136]  5Ø8: ; Eing: (AX) = Adr. Bildschirmspeicher
[573]  5Ø9: ;       (BL) = Attribut
[DF4]  51Ø: ;       (DH,DL) = Zeile, Spalte
[1X4]  511: ;       (DS,SI) = (Segment, Offset)
[W94]  512: ;                  ASCII-Stringende
[EN4]  513: ;                  durch X'ØØ' markiert
[ØØØ]  514:
[U33]  515: zos_1ØØ   proc   near
[Y91]  516:           push   es
[991]  517:           push   di
[AR1]  518:           mov    es,ax
[Y42]  519:           mov    ax,ØØ5Øh
[X61]  52Ø:           mul    dh
[NP1]  521:           xor    dh,dh
[VL1]  522:           add    ax,dx
[7H1]  523:           shl    ax,1
[NP1]  524:           mov    di,ax
[ULØ]  525:           cld
```

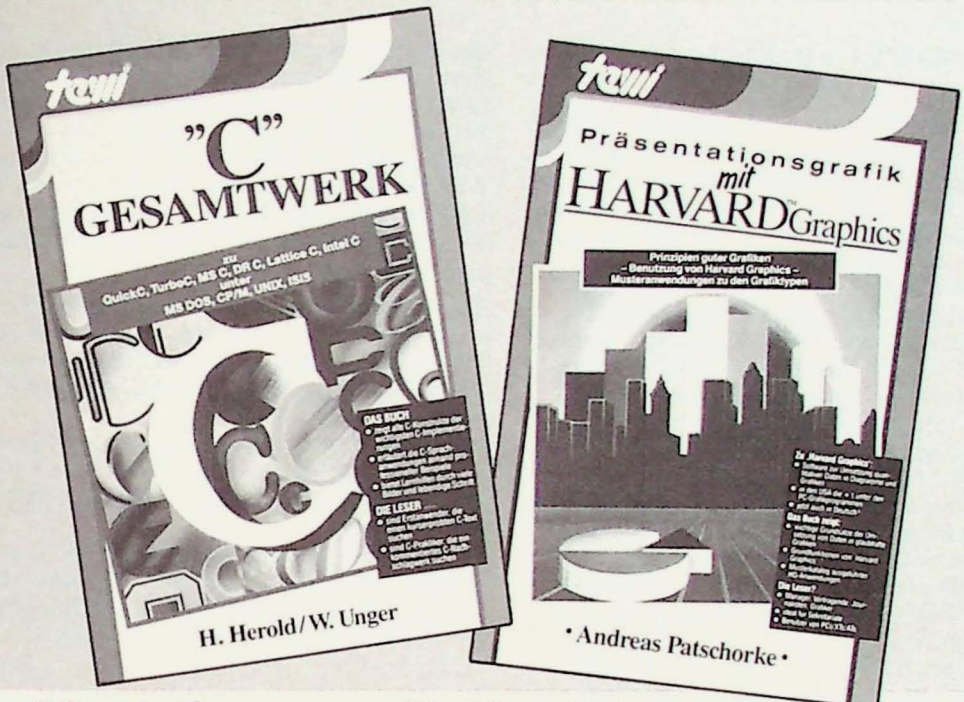
```
[TT2]  526: zos_loop: lodsb
[AP1]  527:           cmp    al,ØØh
[8R1]  528:           je     zos_x
[8P1]  529:           mov    ah,bl
[ZYØ]  53Ø:           stosw
[ZL2]  531:           jmp    zos_loop
[9U2]  532: zos_x:    pop    di
[S71]  533:           pop    es
[3PØ]  534:           ret
[S42]  535: zos_1ØØ   endp
[ØØØ]  536:
[3C4]  537: ;-------------------------------------
[1E6]  538: ; Bildschirm löschen, Segment Adresse
[2G5]  539: ; Bildschirmspeicher ermitteln:
[N42]  54Ø: ; Eing:  keine
[8Y4]  541: ; Ausg:  (AL) = Bildschirm-Modus
[943]  542: ;                3 für Farbe
[ZY3]  543: ;                7 für Monochrom
[HF4]  544: ;        (BX) = Segment-Adresse
[1K4]  545: ;                Bildschirmspeicher
[C34]  546: ;                X'B8ØØ' für Farbe
[YW4]  547: ;                X'BØØØ' für Monochrom
[ØØØ]  548:
[8B3]  549: zst_1ØØ   proc      near
[BL1]  55Ø:           push      bp
[CL1]  551:           push      di
[8N1]  552:           push      si
[HA2]  553:           mov       ax,ØØØ3h
[GK1]  554:           int       1Øh
[AZ1]  555:           mov       ah,15
[GK1]  556:           int       1Øh
[UR1]  557:           cmp       al,7
[8B2]  558:           jne       zst_1Ø1
[FL2]  559:           mov       bx,Øb8ØØh
[ZE2]  56Ø:           jmp       zst_1Øx
[LV3]  561: zst_1Ø1:  mov       bx,Øb8ØØh
[S33]  562: zst_1Øx:  pop       si
[6H1]  563:           pop       di
[5H1]  564:           pop       bp
[3PØ]  565:           ret
[852]  566: zst_1ØØ   endp
[ØØØ]  567:
[LW5]  568: ;=====================================
[364]  569: ; Initialisierungs-Teil:
[ØØØ]  57Ø:
[UØ2]  571: ; Bildschirm
[ØD3]  572: init:     call   zst_1ØØ
[1Y2]  573:         ; prüfen, löschen
[MY2]  574:           mov    kbb_sbsg,bx
[5B3]  575:         ; Cursor ausblenden
[C52]  576:           mov    cx,ØfØØh
[NR1]  577:           mov    ah,Ø1h
[NB1]  578:           int    1Øh
[Ø62]  579:           mov    ax,35Ø9h
[SB1]  58Ø:           int    21h
[RB2]  581:           mov    intØ9,bx
[ØC2]  582:           mov    intØ9+2,es
[6V4]  583:         ; eigenen Bildschirm aufbauen
[9S1]  584:           mov    bl,3Øh
[CX2]  585:           mov    ax,kbb_sbsg
[R42]  586:           mov    dx,ØØØØh
[LW1]  587:           lea    si,l_ØØ
[L32]  588:           call   zos_1ØØ
[LS1]  589:           mov    bl,1bh
[CX2]  59Ø:           mov    ax,kbb_sbsg
[W42]  591:           mov    dx,Ø5ØØh
[2X1]  592:           lea    si,l_Ø5
[L32]  593:           call   zos_1ØØ
[9S1]  594:           mov    bl,3Øh
[CX2]  595:           mov    ax,kbb_sbsg
[C52]  596:           mov    dx,15Ø4h
[A62]  597:           lea    si,l_211
[L32]  598:           call   zos_1ØØ
[9S1]  599:           mov    bl,3Øh
[CX2]  6ØØ:           mov    ax,kbb_sbsg
[D52]  6Ø1:           mov    dx,16Ø4h
[D62]  6Ø2:           lea    si,l_221
[L32]  6Ø3:           call   zos_1ØØ
[MS1]  6Ø4:           mov    bl,1ch
[CX2]  6Ø5:           mov    ax,kbb_sbsg
[F52]  6Ø6:           mov    dx,18Ø4h
[L62]  6Ø7:           lea    si,l_241
[L32]  6Ø8:           call   zos_1ØØ
[ØØØ]  6Ø9:
[HU5]  61Ø: ; eigenes Tastatur-Unterbrechungs-
[WH3]  611: ; Programm aktivieren
[ØØØ]  612:
[BV3]  613:           mov    dx,offset myintØ9
[W52]  614:           mov    ax,25Ø9h
[SB1]  615:           int    21h
[GU1]  616:           jmp    dloop
[ØØØ]  617:
[LW5]  618: ;=====================================
[ØØØ]  619:
[BW1]  62Ø: code      ends
[C72]  621: end       start
```

Listing. Tastatur und Tastaturtreiber – mit »tastmon.asm« enträtselt

tewi tewi tewi

tewi Verlag GmbH · Theo-Prosel-Weg 1

8000 München 40 · Telefon 089/1292090



C-Gesamtwerk
Herold/Unger
576 S., DM 79,–
zu Quick C, Turbo C, DR C,
Lattice C, Intel C unter
MS DOS, CP/M, UNIX, ISIS.

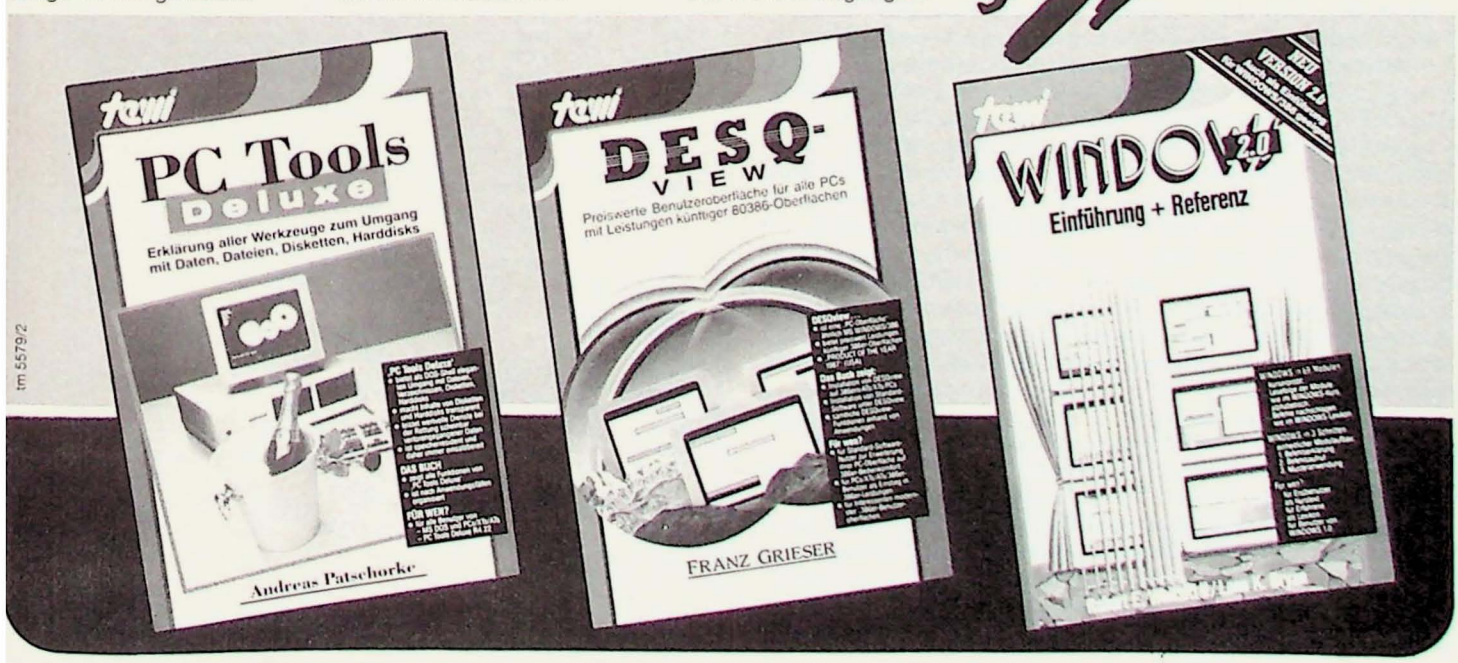
Präsentationsgrafik mit
HARVARD Graphics
Patschorke, 172 S., DM 59,–
Gute Zahlengrafik und ihre
Erstellung auf dem PC mit
HARVARD Graphics, deutsche
Version. Mustergrafiken.

PC Tools Deluxe
Patschorke,
160 S., DM 49,–
Moderne Alternative zu
NORTON. Alle Funktionen
nach typischen Anwen-
dungsfällen organisiert.

DESQview
Grieser, 336 S., DM 59,–
Problemlos DOS-Programme
unter Multitasking-Oberfläche
integrieren. Bestbeurteilter
WINDOWS-Konkurrent.

WINDOWS 2.0
Einführung + Referenz
Whitsitt/Bryan
496 S., DM 79,–
Auch als Einführung für
WINDOWS/386 geeignet.

DRUCKFRISCH!



tm 5579/2

Für Eilbestellungen: Hotline 02191-342077

Tips und Tricks

Auch in der DOS-Extra, als Sonderausgabe der DOS International, dürfen natürlich die Tips und Tricks nicht fehlen. Auf den folgenden Seiten finden Sie deshalb einige ausgewählte Tools und Utilities in Turbo Pascal, C, GW-Basic und Assembler. Alle Quelltexte sind gut dokumentiert, und viele Routinen eignen sich zum Einbau in Ihre selbstgeschriebenen Programme.

Manipulation des Environment-Blocks

Eine Stapeldatei wird durch Tastaturabfragen erst so richtig komfortabel. Zwar gibt es zu diesem Zweck verschiedene Hilfsprogramme, wie zum Beispiel »ask« aus den Norton Utilities, aber diese sind meist nicht sehr flexibel. Es sind nur einbuchstabige Eingaben und vordefinierte Abfragen erlaubt. Das hier vorgestellte Programm »input.pas« (Listing 1) ist wesentlich flexibler und erlaubt fast beliebige Eingaben innerhalb einer Stapeldatei. Sie können einzelne Tasten oder auch beliebig lange Texte abfragen, die der Anwender mit [Enter] abschließt. Zusätzlich sind die Ausgabe eines Hinweistextes sowie einer Eingabeaufforderung und die Positionierung der Eingabe auf dem Bildschirm möglich.

Das Programm ist in Turbo Pascal 4.0 geschrieben und setzt die vom Anwender vorgenommenen Eingaben direkt in den MS-DOS-Umgebungsbereich, den »Environment-Block«. Der Environment-Block besteht aus einer Folge von Zeichenketten, die jeweils durch den ASCII-Wert 0 getrennt und durch zwei Nullen abgeschlossen sind. Diese Strings besitzen das Format »name = xxx« und enthalten zum Beispiel Informationen über Pfad und Verzeichnis, in dem sich der Kommandoprozessor »command .com« befindet. Sie können sich den Inhalt des Environment-Blocks ansehen, indem Sie den MS-DOS-Befehl »set« eingeben.

Das Problem beim direkten Zugriff auf den Environment-Block von einem Programm aus liegt darin, daß dieser je nach MS-DOS-Version, Anzahl und Art von eventuell installierten Gerätetreibern an einer anderen Speicherstelle zu finden ist.

Die Prozedur »GetParameter« im Listing 1 stellt zunächst fest, ob Eingabeaufforderung, Variablenname und Parameter gewünscht sind. Die Parameter sind im Listing erklärt. Bei der Eingabeaufforderung ist zu beachten, daß Turbo Pascal Leerzeichen als Trennungszeichen zwischen den Kommandozeilenparametern versteht, deshalb müssen Sie Leerzeichen innerhalb des auszugebenden Textes durch das ASCII-Zeichen 255 ([Alt-255]) ersetzen. Haben Sie das Programm ohne Parameter aufgerufen, dann wird ein Hilfstext auf dem Bildschirm ausgegeben. Es folgt die Tastaturabfrageroutine, die mit Hilfe der Funktion »Caps_String« in Großbuchstaben umgewandelt wird.

Der nächste Programmabschnitt ermittelt die Adresse des Environment-Blocks. Wenn Sie von MS-DOS aus ein Programm aufrufen, dann setzt das Betriebssystem an den Anfang des zugewiesenen Speichers ein sogenanntes »Program Segment Prefix« (PSP). Dieses enthält unter anderem eine Kopie des MS-DOS-Umgebungsbereichs. In der Speicherstelle 2Chex ist die Segmentadresse des Environment-Blocks als 16-Bit-Wert gespeichert, wobei der Offset immer bei 0 liegt. Der Anfang des für den Kommandoprozessor reservierten Bereichs ist über den undokumentierten Interrupt ab Adresse BAhex zu erfahren, der immer auf den Anfang des Kommandoprozessors zeigt.

Das Programm »input« liest nun den Environment-Block und prüft, ob der vom Anwender gewünschte Variablenname noch nicht vorhanden ist. Danach wird die Größe des Environment-Bereichs festgestellt. Das Ende dieses Bereichs liegt immer auf geraden Speicher-Paragraphen (glatt durch 16 teilbare Adressen), und das erste Zeichen im nächsten Paragraphen ist ein »M«. Wenn die Größe nicht ausreicht, folgt eine Fehlermeldung.

Ab MS-DOS 3.0 können Sie den standardmäßig auf 160 Byte eingestellten Bereich vergrößern. Setzen Sie dazu in die Systemdatei »config.sys« den Befehl:

```
shell=command.com /p /e:xxxx
```

Für »xxxx« geben Sie die gewünschte Größe des Environment-Bereichs in Byte ein. Diese kann zwischen 128 und 32768 Byte liegen. Durch den Parameter »/p« wird der Kommandoprozessor resident geladen und die Datei »autoexec.bat« ausgeführt.

Nachdem »input.pas« die Größe des Environment-Bereichs überprüft hat, schreibt es den Variablennamen mit dem vom Benutzer eingegebenen Inhalt in den Arbeitsspeicher.

Zum Schluß noch ein Hinweis zum Testen des Programms: Es funktioniert nur als eigenständiges, compiliertes Programm und nicht über den MS-DOS-Ausgang der Turbo-Pascal-Entwicklungsumgebung.

(Karsten Hoffmann/tr)

```
[CØ3]    1: { Name:  input.pas
[LH2]    2:          Turbo Pascal 4.Ø
[Z16]    3:          Eingabeprogramm für Stapeldateien
[AØ4]    4:   Autor: Karsten Hoffmann
[TL4]    5:          Copyright Everts & Hagedorn
[833]    6:          DOS International
[U7Ø]    7: }
[5H1]    8: uses crt;
[ØØØ]    9:
[6XØ]   1Ø: const
[TU5]   11: Meldung     : array[1..18] of string =
[A76]   12: (#13#1Ø'INPUT von Karsten Hoffmann'#13#1Ø,
[DH6]   13: 'Eingabeprogramm für Stapeldateien.',
[9E7]   14: 'Ermöglicht die Ausgabe einer Meldung und',
[836]   15: 'eine Tastaturabfrage, die im MS-DOS-',
[9Y6]   16: 'Umgebungs-Bereich gespeichert wird, und',
[8A7]   17: 'sich über IF %VAR% == ... abfragen läßt.',
[S86]   18: 'Wurde der Parameter /R angegeben, ist',
[4E6]   19: 'eine beliebig lange Eingabe möglich,',
[XD6]   2Ø: 'sonst wird nur eine Taste abgefragt.',
[6D6]   21: 'Die Eingabe wird in Großbuchstaben',
[ST4]   22: 'umgewandelt. Der Parameter',
[645]   23: '/L:Zeile,Spalte bewirkt eine',
[WR3]   24: 'Positionierung.'#13#1Ø,
[VLC]   25: 'Syntax: INPUT Variable [/E:Eingabeaufforde
            rung] [/R] [/L:Zeile,Spalte]',
[F97]   26: #13#1Ø'Parameter müssen durch Leerzeichen',
[B66]   27: 'getrennt werden, Leerzeichen in der',
[Y77]   28: 'Eingabeaufforderung werden als [ALT 255]',
[VD2]   29: 'eingegeben.');
[ØØØ]   3Ø:
[RNØ]   31: var
[512]   32: i           : Byte;
[833]   33: Eingabe     : String;
[563]   34: Variable    : String;
[ØØØ]   35:
[EB7]   36: function Caps_String(Str:String):String;
[N38]   37: {wandelt einen String in Großbuchstaben um}
[RNØ]   38: var
[BT2]   39: Buchst          : Char;
[8R3]   4Ø: NeuString       : String;
[MA2]   41: i               : Byte;
[ØØØ]   42:
[MUØ]   43: begin
[1E2]   44: NeuString := '';
```

Listing 1. »input.pas«, eine Eingaberoutine für Stapeldateien

```
[R84]   45: for i := 1 to Length(Str) do
[MUØ]   46:     begin
[U84]   47:     Buchst := UpCase(Str[i]);
[DC5]   48:     NeuString := NeuString + Buchst
[ZNØ]   49:     end;
[3C4]   5Ø: Caps_String := NeuString;
[ZNØ]   51: end;
[ØØØ]   52:
[ØØØ]   53:
[744]   54: procedure Getparameter;
[YS6]   55: {holt Parameter und fragt Tastatur ab}
[RNØ]   56: var
[ØE3]   57: ParamZeile : String;
[EW2]   58: Param      : String;
[512]   59: i          : Byte;
[CT2]   6Ø: Eing       : String;
[7Y2]   61: Enter      : boolean;
[TX2]   62: Zeile      : Integer;
[T13]   63: Spalte     : Integer;
[4U2]   64: Code       : Integer;
[WF2]   65: Z          : String;
[EF2]   66: S          : String;
[TE2]   67: Komma      : Byte;
[ØØØ]   68:
[MUØ]   69: begin
[NM1]   7Ø: Eing := '';
[EB2]   71: Variable := '';
[DB4]   72: for i := 1 to ParamCount do
[MUØ]   73:    begin
[KØ7]   74:    Param := Caps_String(Copy(ParamStr(i),1,
            2));
[DW2]   75:    if Param = '/R' then
[N42]   76:       Enter := true
[ZW3]   77:    else if Param = '/E' then
[XV4]   78:       Eing := Copy(ParamStr(i),4,255)
[VX3]   79:    else if Param = '/L' then
[MUØ]   8Ø:       begin
[P94]   81:       Z := Copy(ParamStr(i),4,255);
[BX2]   82:       Komma := Pos(',',Z);
[BR3]   83:       S := Copy(Z,Komma+1,255);
[8L3]   84:       Z := Copy(Z,1,Komma-1);
[D13]   85:       Val(Z,Zeile,Code);
[U83]   86:       Val(S,Spalte,Code);
[LM3]   87:       gotoxy(spalte,Zeile);
[4MØ]   88:       end
[ZRØ]   89:    else
[3A6]   9Ø:       variable := Caps_String(ParamStr(i));
[ZNØ]   91: end;
[ØØØ]   92:
[BY1]   93: write(Eing);
[B53]   94: if Enter = true then
[KF2]   95:    Readln(Eingabe)
[ZRØ]   96: else
[643]   97:    Eingabe := ReadKey;
[Z75]   98: Eingabe := Caps_String(Eingabe);
[BT5]   99: Eingabe := Variable + '=' + Eingabe + #Ø;
[ZNØ]  1ØØ: end;
[ØØØ]  1Ø1:
[EG4]  1Ø2: procedure SetEnvironment;
[RY6]  1Ø3: {schreibt die Abfrage in das Environment}
[6XØ]  1Ø4: const
[DY1]  1Ø5: M              = 77;
[BW6]  1Ø6: {'M' Kenzeichen für Environment-Ende}
[ØØØ]  1Ø7:
[RNØ]  1Ø8: var
[413]  1Ø9: Zaehler        : Byte;
[K13]  11Ø: EnvEnde        : Word;
[TD3]  111: EnvAlt         : String;
[BX2]  112: i              : Integer;
[Ø43]  113: AnfangDOS      : Word;
[MK3]  114: Environment    : Word;
[BM2]  115: Ende           : Byte;
[BS3]  116: EnvLaenge      : Integer;
[CZ3]  117: AnfVariable    : Integer;
[XF3]  118: EnvAlt1        : String;
[UP3]  119: LaengeNeu      : Integer;
[VD3]  12Ø: EnvNeu         : String;
[ØØØ]  121:
[MUØ]  122: begin
[LZ1]  123: Zaehler := Ø;
[EN2]  124: EnvEnde := $FFFF;
[EV1]  125: EnvAlt := '';
[PYØ]  126: i := 1;
[WE1]  127: Ende := Ø;
[V24]  128: AnfangDOS := MemW[$Ø:$BA];
[AK5]  129: {Undokumentierter Interrupt,}
[336]  13Ø: Environment := MemW[AnfangDOS:$2C];
[Y96]  131: {zeigt auf den Beginn von COMMAND.COM}
[5F3]  132: while (EnvEnde <> Ø) do
[F84]  133: {Environment auslesen}
[MUØ]  134:   begin
[BU2]  135:   EnvAlt := EnvAlt +
[HE5]  136:            Chr(Mem[Environment:Zaehler]);
[BK6]  137:   EnvEnde := MemW[Environment:Zaehler];
[H62]  138:   Inc(Zaehler);
[ZNØ]  139:   end;
[ØØØ]  14Ø:
[431]  141: i := 144;
[HV2]  142: while (Ende <> M) do
[BM5]  143: {'M' bedeutet Environment-Ende}
[MUØ]  144:   begin
[RN3]  145: {gerader Paragraph}
[3E1]  146:   Inc(i,16);
[UK6]  147: {Größe des Environments bestimmen}
[1W4]  148:   Ende := Mem[Environment:i];
[ZNØ]  149:   end;
[ØØØ]  15Ø:
[DV2]  151: EnvLaenge := i -1 ;
[P22]  152: EnvAlt1 := '';
[ØØØ]  153:
[CD6]  154: AnfVariable := Pos(Variable+'= ,EnvAlt);
[KN3]  155: if AnfVariable <> Ø then
[MUØ]  156:    begin
[HH6]  157:    EnvAlt1 := Copy(EnvAlt,1,AnfVariable-1);
[V25]  158:    Delete(EnvAlt,1,AnfVariable-1);
[RT5]  159:    EnvAlt := Copy(EnvAlt,Pos(#Ø,EnvAlt)
[LXØ]  16Ø:                  +1,255);
[ZNØ]  161:    end;
[ZA7]  162: LaengeNeu := Length(EnvAlt1)+Length(EnvAlt)
[C53]  163:              +Length(Eingabe)+1;
[ØØØ]  164:
[XN4]  165: if LaengeNeu > EnvLaenge then
[GA2]  166:    Writeln(#13#1Ø+
[T95]  167:            'Environmentbereich zu klein');
[ØØØ]  168:
[LN5]  169: EnvNeu := EnvAlt1+EnvAlt+Eingabe+#Ø;
[R14]  17Ø: for i := 1 to LaengeNeu do
[X14]  171:     Mem[Environment:i-1] :=
[XG5]  172:         Mem[Seg(EnvNeu):Ofs(EnvNeu)+i];
[ØØØ]  173:
[7V5]  174: for i := LaengeNeu+1 to EnvLaenge do
[T24]  175:     Mem[Environment:i] := Ø;
[ZNØ]  176: end;
[ØØØ]  177:
[ØØØ]  178:
[663]  179: {HAUPTPROGRAMM}
[MUØ]  18Ø: begin
[UF3]  181: if Paramcount = Ø then
[MUØ]  182:    begin
[DB1]  183:    Clrscr;
[1R2]  184:    for i := 1 to 18 do
[UK3]  185:        Writeln(Meldung[i]);
[4MØ]  186:    end
[ZRØ]  187: else
[MUØ]  188:    begin
[892]  189:    Getparameter;
[3X2]  19Ø:    SetEnvironment;
[ZNØ]  191:    end;
[KNØ]  192: end.
```

Listing 1. »input.pas«, eine Eingaberoutine für Stapeldateien

Beliebig lange Pfadangaben

Im vorhergehenden Beitrag wurde die direkte Manipulation des Environment-Blocks beschrieben. Dadurch läßt sich auch ein anderes Problem lösen: die Begrenzung von Eingabezeilen unter MS-DOS auf maximal 127 Zeichen. Wer zum Beispiel längere Pfadangaben benötigt, mußte bisher die Namen seiner Unterverzeichnisse kürzen.
Genau an dieser Stelle setzt das Programm »xpath.pas« (Listing 2) an, das sich die MS-DOS-Pfadangabe nicht über eine Eingabezeile, sondern aus einer Textdatei holt. Deren Inhalt darf fast beliebig viele, nur durch die maximale Größe des Environment-Bereichs begrenzte Zeichen enthalten und wird als neuer Pfad direkt in den Umgebungsbereich des Kommandointerpreters geschrieben. Die wesentlichen Funktionen und Routinen des Programms stimmen mit »input.pas« (Listing 1) überein.
Die Anwendung ist sehr einfach: Legen Sie sich (zum Beispiel mit »copy con datei.exe«) eine beliebige Textdatei an, und schreiben Sie dort Ihren gewünschten Pfad hinein. Dann rufen Sie »xpath« auf und geben als Parameter den Namen dieser Datei und eventuell einen Verzeichnispfad auf die Datei mit an. Beispiel:

```
copy con pfad.txt
c:\tools\pctools\bin;c:\tools\pctools;c:\tools;c:\dos\drucker;c:\dos\bin;c:\dos;c:\texte\firma;c:\texte\privat;c:\texte\meier;c:\word;c:\word\bin;
↑Z
```

Wichtig ist, daß Sie keine Leerzeichen innerhalb der neuen Pfadangabe verwenden. Starten Sie dann »xpath« mit

```
xpath pfad.txt
```

und der neue MS-DOS-Pfad ist aktiv.

(Karsten Hoffmann/tr)

```
[7X2]    1: {  Name: xpath.pas
[LH2]    2:          Turbo Pascal 4.Ø
[DE5]    3:          Setzen eines Pfades mit mehr als
[3X1]    4:            127 Zeichen.
[AØ4]    5:    Autor: Karsten Hoffmann
[TL4]    6:          Copyright Everts & Hagedorn
[833]    7:          DOS International
[U7Ø]    8: }
[DG2]    9: program XPATH;
[ØØØ]   1Ø:
[6XØ]   11: const
[ØØ8]   12: Meldung1 = #13#1Ø'XPATH  von Karsten Hoffma
            nn'#13#1Ø;
[LAE]   13: Meldung2 = 'XPATH setzt den MS-DOS-Pfad, de
            r in einer ASCII-Datei gespeichert ist.'#13
            #1Ø;
[CB7]   14: Meldung3 = 'Aufruf: XPATH [LW:][Pfad]Datei'
            ;
[MK7]   15: Meldung4 = #1Ø'Environment-Bereich zu klein
            .'#1Ø;
[CP6]   16: Meldung5 = #13#1Ø'Der Pfad ist zu lang.'#13
            #1Ø;
[A17]   17: Meldung6 = #13#1Ø'Datei nicht gefunden.'#13
            #1Ø;
[ØØØ]   18:
[X53]   19: procedure Setpath;
[ØØØ]   2Ø:
[6XØ]   21: const
[DØ1]   22: M = 77;
[617]   23: {'M' Kennzeichen für Environment-Ende}
[ØØØ]   24:
[RNØ]   25: var
[LZ2]   26: Datei            : Text;
[WA4]   27: PathEing,EnvAlt,PathNeu,
[9W2]   28:                  Zeichen : String;
[GU3]   29: AnfangDOS,Environment,
[P14]   3Ø:         EnvEnde,Zaehler : Word;
[PF4]   31: EnvLaenge,i             : Integer;
[EP2]   32: PathAnfang,Ende,
[3U2]   33:              LaengeNeu : Byte;
[ØØØ]   34:
[MUØ]   35: begin
[U74]   36: Assign(Datei,ParamStr(1));
[DXØ]   37:  {$I-}
[F52]   38:  Reset(Datei);
[BXØ]   39:  {$I+}
[5A3]   4Ø:  if IOResult <> Ø then
[AT5]   41: {prüfen, ob Datei vorhanden ist}
[MUØ]   42:     begin
[ZS4]   43:        Writeln(Meldung6,Meldung3);
[RPØ]   44:        Halt
[4MØ]   45:     end
[ZRØ]   46:  else
[LZ1]   47:     Zaehler := Ø;
[EN2]   48:     EnvEnde := $FFFF;
[EV1]   49:     EnvAlt := '';
[PYØ]   5Ø:     i := 1;
[WE1]   51:     Ende := Ø;
[2V3]   52:     Readln(Datei,PathEing);
[V24]   53:     AnfangDOS := MemW[$Ø:$BA];
[AK5]   54: {Undokumentierter Interrupt,}
[336]   55:     Environment := MemW[AnfangDOS:$2C];
[Y96]   56: {zeigt auf den Beginn von COMMAND.COM}
[5F3]   57:     while (EnvEnde <> Ø) do
[F84]   58: {Environment auslesen}
[MUØ]   59:        begin
[NE1]   6Ø:          EnvAlt :=
[ZT6]   61:     EnvAlt + Chr(Mem[Environment:Zaehler]);
[SP1]   62:          EnvEnde :=
[YS4]   63:     MemW[Environment:Zaehler];
[H62]   64:          Inc(Zaehler);
[ZNØ]   65:        end;
[ØØØ]   66:
[431]   67:     i := 144;
[HV2]   68:     while (Ende <> M) do
[BM5]   69: {'M' bedeutet Environment-Ende}
[MUØ]   7Ø:        begin
[RN3]   71: {gerader Paragraph}
[3E1]   72:          Inc(i,16);
[UK6]   73: {Größe des Environments bestimmen}
[1W4]   74:          Ende := Mem[Environment:i];
[ZNØ]   75:        end;
[ØØØ]   76:
[DV2]   77:     EnvLaenge := i -1 ;
[3B5]   78:     PathAnfang := Pos('PATH=',EnvAlt);
[YE1]   79:     Ende := 1;
[LYØ]   8Ø:     i := Ø;
[MU2]   81:     while (Ende <> Ø) do
[DP4]   82: {löscht den alten Pfad}
[MUØ]   83:        begin
[B51]   84:          Ende :=
[KS5]   85:             Mem[Environment:PathAnfang+4+i];
[951]   86:          Inc(i);
[ZNØ]   87:        end;
[XØ5]   88:     Delete(EnvAlt,PathAnfang+5,i-1);
[ØØØ]   89:
[FY2]   9Ø:     if Length(EnvAlt) +
[R84]   91:                Length(PathEing) > 254 then
[F53]   92:        Writeln(Meldung5);
[ØØØ]   93:
[686]   94:     Insert(PathEing,EnvAlt,PathAnfang+5);
[GX3]   95:     PathNeu := EnvAlt+Chr(Ø);
[P64]   96: {neue Pfad hinzufügen}
[ØØØ]   97:
[ZM5]   98:     if Length(PathNeu) > EnvLaenge then
[MUØ]   99:        begin
[E53]  1ØØ:          Writeln(Meldung4);
[TV4]  1Ø1: {Environment groß genug?}
[DUØ]  1Ø2:          Halt;
[ZNØ]  1Ø3:        end;
[ØØØ]  1Ø4:
[7X4]  1Ø5:     for i := 1 to Length(PathNeu) do
[X14]  1Ø6:        Mem[Environment:i-1] :=
[5N5]  1Ø7:          Mem[Seg(PathNeu):Ofs(PathNeu)+i];
[KK6]  1Ø8: {Environment in Speicher schreiben}
[EM4]  1Ø9:     for i := Length(PathNeu)+1 to
[H12]  11Ø:                                EnvLaenge do
[YØ4]  111:        Mem[Environment:i] := Ø
[ZNØ]  112: end;
[ØØØ]  113:
[663]  114: {HAUPTPROGRAMM}
[MUØ]  115: begin
[UF3]  116: if Paramcount = Ø then
[MUØ]  117:    begin
[U56]  118:       Writeln(Meldung1,Meldung2,Meldung3);
[4MØ]  119:    end
[ZRØ]  12Ø: else
[EF1]  121:    Setpath;
[KNØ]  122: end.
```

Listing 2. »xpath.pas« erlaubt Pfadangaben mit mehr als 127 Zeichen

Ein MS-DOS-Filter in C

Obgleich C eine Hochsprache ist, so erlaubt sie trotzdem sehr hardware- und systemnahe Programme. Beispielsweise können Sie – viel einfacher als etwa in Turbo Pascal – Interrupts direkt aufrufen. Daneben lassen sich aber auch Funktionen, wie das direkte Schreiben in den Bildschirmspeicher, ohne eine einzige Assemblerzeile realisieren.

Andererseits sind C-Programme sehr schön in die MS-DOS-Umgebung eingebettet. So werden für Ein- und Ausgabefunktionen meist »saubere« Systemaufrufe benutzt, das Umleiten von Ein- und Ausgaben auf Betriebssystemebene durch Pipes (»dir more«) und Filter ist somit jederzeit möglich. Als Beispiel für einen nützlichen Filter dient zunächst das Programm »ext.c«, das die Ausgabe des Dir-Befehls durchsucht.

Bei langen und unübersichtlichen Verzeichnissen ergibt sich meist eine mühselige Suche nach allen ausführbaren Programmen (mit den Endungen »exe«, »com« und »bat«). Ein erster Schritt zur Lösung dieses Problems könnte eine Stapeldatei sein, die das vorhandene Filterprogramm »find.exe« des Betriebssystems nutzt und aus der Verzeichnisliste alle (und nur die) vorhandenen Programmdateien auflistet. Die Stapeldatei »pdir.bat« finden Sie in Listing 3.

Hier werden alle Bildschirmausgaben zunächst gefiltert und dann in eine Datei umgeleitet. Nach mehrmaligem Aufruf der MS-DOS-Hilfsprogramme enthält die Textdatei schließlich die Angaben zu allen Programmdateien und die üblichen Informationen über das Disketten-Label und die Anzahl der gefundenen Dateien. Da hier das Dir-Kommando und die Filter mehrfach aufgerufen werden müssen, arbeitet das Programm sehr langsam.

Der nächste Schritt zu einer etwas eleganteren Lösung ist ein C-Programm. Dieses Programm holt sich alle Eingaben vom Standard-Eingabegerät, so daß es die umgeleitete Ausgabe des Dir-Befehls auswerten kann. Die Namenserweiterungen der verschiedenen Dateitypen werden als Parameter in der Kommandozeile übergeben. Folgende Angaben gibt das Programm anschließend aus:

- Name des Datenträgers
- Name, Größe und Datum der gesuchten Dateien
- Gesamtanzahl der Dateien und freier Speicherplatz

Wie lassen sich nun alle wichtigen Informationen aus der Ausgabe des Dir-Befehls erkennen? Was ist bei der Auswahl der angeforderten Daten zu beachten?
Zunächst fällt auf, daß alle Systemmeldungen Kleinbuchstaben enthalten, was bei Dateinamen nicht vorkommt. Es müssen also alle Textzeilen mit Kleinbuchstaben ungefiltert ausgegeben werden, sie enthalten zusätzliche Informationen über den jeweiligen Datenträger.
Die Namenserweiterungen aller Dateinamen befinden sich in jeder Zeile in den Spalten 10 bis 13. Hier muß also überprüft werden, ob die Datei die gesuchte Erweiterung hat, also angezeigt werden darf oder nicht.
Die Anzahl der gefundenen Dateien muß ausgegeben werden. Sie läßt sich durch Zählen der gültigen Dateien ermitteln.
Systemmeldungen können je nach MS-DOS-Version deutsch oder englisch sein. Anstelle der Namenserweiterung in den Spalten 10 bis 13 enthält die letzte Zeile die drei Zeichen (Leerzeichen mitgezählt) » Da« (»Datei«) oder » Fi« (»File«). Sie muß zusammen mit der korrekten Anzahl der gefundenen Dateien ausgegeben werden.
Das Programm »ext.c« (Listing 4) arbeitet genau nach diesen Anforderungen. Somit sieht die Bildschirmausgabe optisch genauso aus wie das Dir-Kommando ohne jeden Filter. Lediglich die Anzahl der freien Bytes stimmt nicht ganz, weil die temporär vorhandene Eingabedatei ebenfalls Platz beansprucht.
Die genaue Schreibweise des Ext-Befehls lautet wie folgt:

```
dir| ext [Typ1] [Typ2] ...
```

Fehlen die Parameter, dann zeigt das Programm einen Hilfstext. Dies ist deshalb sinnvoll, weil das Programm normalerweise auf eine Eingabe über die Tastatur wartet. Wird es jedoch ohne Pipe-Zeichen und Parameter aufgerufen, dann hängt es sich scheinbar auf, da ein Dateiendezeichen »^Z« erwartet wird. Wie soll ein nichtsahnender Benutzer, der nur noch den Cursor und einen leeren Bildschirm sieht, dies wissen?
Die Länge jeder Eingabezeile ist auf ein Maximum von 127 Zeichen gesetzt. Das entspricht der maximalen Länge einer Kommandozeile des Betriebssystems. Eine Verzeichnis-Anzeige ist nur 40 Stellen lang, also sollte dieser Wert ausreichen. Falls aber durch einen Bedienungsfehler eine längere Zeile übergeben würde (vielleicht »dir /w| ext ...«), dann könnte das Programm wirklich abstürzen.
Die Batchdatei »pdir.bat« sieht jetzt viel übersichtlicher aus:

```
@echo off
dir| ext com exe bat
```

Natürlich können Sie »ext.c« auch dazu einsetzen, um zum Beispiel alle Textdateien mit den Endungen »txt«, »doc« und »asc« aus einem Verzeichnis zu filtern.

(Uwe Sawallisch/tr)

```
[FD1]    1: @ECHO OFF
[T34]    2: DIR | FIND/v "   " > DATEIEN
[PR3]    3: IF EXIST *.COM GOTO COM
[HR3]    4: IF EXIST *.EXE GOTO EXE
[YP3]    5: IF EXIST *.BAT GOTO BAT
[HØ6]    6: ECHO Keine Programmdateien >> DATEIEN
[FH1]    7: GOTO ENDE
[ØØØ]    8:
[6PØ]    9: :COM
[G65]   1Ø: DIR *.COM | FIND " COM" >> DATEIEN
[HR3]   11: IF EXIST *.EXE GOTO EXE
[YP3]   12: IF EXIST *.BAT GOTO BAT
[6H1]   13: GOTO FIND
[ØØØ]   14:
[8RØ]   15: :EXE
[D65]   16: DIR *.EXE | FIND " EXE" >> DATEIEN
[YP3]   17: IF EXIST *.BAT GOTO BAT
[6H1]   18: GOTO FIND
[ØØØ]   19:
[KMØ]   2Ø: :BAT
[465]   21: DIR *.BAT | FIND " BAT" >> DATEIEN
[ØØØ]   22:
[ØUØ]   23: :FIND
[PE4]   24: FIND/c "   " DATEIEN >> DATEIEN
[ØØØ]   25:
[9UØ]   26: :ENDE
[332]   27: TYPE DATEIEN
[Ø12]   28: DEL DATEIEN
```

Listing 3. Die Stapeldatei »pdir.bat« zeigt alle ausführbaren Programme in einem Verzeichnis

```
[9F4]    1: /*---------------------------------------
[5L3]    2:  * Programmdatei EXT.C
[WY5]    3:  * Filter zum Lesen des Directorys: Es
[P66]    4:  * werden nur die als Parameter überge-
[SK4]    5:  * benen Dateitypen gezeigt.
[M2Ø]    6:  *
[B95]    7:  *    Syntax: DIR | EXT EXE COM z.B.
[M2Ø]    8:  *
[EC4]    9:  *---------------------------------------
[CM5]   1Ø:  * Autor: Uwe Sawallisch ***************
[9F4]   11:  *---------------------------------------*/
[ØØØ]   12:
[HW2]   13: #include "stdio.h"
[8V2]   14: #include "ctype.h"
[ØØØ]   15:
[SC2]   16: #define FALSE Ø
[RD2]   17: #define TRUE  1
[RD2]   18: #define LEN 129
[ØØØ]   19:
[GL4]   2Ø: /*---------------------------------------*/
[5G2]   21: main(argc,argv)
[TH1]   22: int argc;
[R42]   23: char *argv[];
[P7Ø]   24: {
[PM5]   25:    char dirline[LEN],filetyp[LEN];
[1W2]   26:    int i,filezahl=Ø;
[ØØØ]   27:
[W95]   28:    if(argc==1)          /* Fehlermeldung */
[P7Ø]   29:    {
[6T2]   3Ø:       fprintf(stderr,
[FW4]   31:               "Syntax: DIR | EXT [typ]\n");
[6T2]   32:       fprintf(stderr,
[M45]   33:         "   z.B. DIR | EXT com exe bat\n");
[C61]   34:       exit(1);
[U7Ø]   35:    }
[ØØØ]   36:
[X76]   37:    for(i=1;i<argc;i++) /* Parameter groß*/
[AV2]   38:        strupr(argv[i]);
[ØØØ]   39:
[4KØ]   4Ø:    i=Ø;
[DØ2]   41:    printf("\n");
[ØØØ]   42:
[226]   43:    while(!feof(stdin)) /* Eingaben lesen*/
[P7Ø]   44:    {
[5C2]   45:       gets(dirline);
[ØØØ]   46:
[C73]   47:       if(!strlen(dirline))
[AM1]   48:          continue;
[ØØØ]   49:
[HX3]   5Ø:       if(islower(dirline[2]))
[HU3]   51:          printf("%s\n",dirline);
[ZRØ]   52:       else
[P7Ø]   53:       {
[XG4]   54:          substr(filetyp,dirline,9,3);
[Ø85]   55:          if(filecheck(filetyp,argc,argv))
[P7Ø]   56:          {
[2W1]   57:             filezahl++;
[TP3]   58:             printf("\n%s",dirline);
[U7Ø]   59:          }
[X25]   6Ø:          else if(!(strcmp(filetyp," Da") &&
[SN3]   61:                   strcmp(filetyp," Fi")))
[P7Ø]   62:          {
[DT4]   63:             substr(filetyp,dirline,1Ø,29);
[Z15]   64:             printf("\n%9d %s\n\n",filezahl,
[EL1]   65:                    filetyp);
[PØ1]   66:             break;
[U7Ø]   67:          }
[U7Ø]   68:       }
[UFØ]   69:       i++;
[U7Ø]   7Ø:    }
[U7Ø]   71: }
[GL4]   72: /*---------------------------------------*/
[AX3]   73: /* Substring kopieren: */
[ER4]   74: substr(aus,ein,anfang,laenge)
[VY2]   75: char aus[],ein[];
[913]   76: int anfang,laenge;
[P7Ø]   77: {
[5B1]   78:    int i,j;
[ØØØ]   79:
[MC5]   8Ø:    for(i=anfang,j=Ø;j<laenge;i++,j++)
[8N2]   81:       aus[j]=ein[i];
[ØØØ]   82:
[DL1]   83:    aus[j]=Ø;
[U7Ø]   84: }
[GL4]   85: /*---------------------------------------*/
[324]   86: /* Anzeige oder nicht? */
```

Listing 4. »ext.c«, ein MS-DOS-Filter, der aus einer Verzeichnisausgabe alle Programmdateien heraussucht

```
[S45]    87: filecheck(extension,argc,argv)
[723]    88: char extension[];
[TH1]    89: int argc;
[R42]    90: char *argv[];
[P70]    91: {
[6C3]    92:     int i,retval=FALSE;
[000]    93:
[MY2]    94:     for(i=1;i<argc;i++)
[DK3]    95:         if(!strncmp(argv[i],
[FU1]    96:                     extension,
[ZV2]    97:                     strlen(argv[i])))
[T52]    98:             retval=TRUE;
[000]    99:
[PN2]   100:     return(retval);
[U70]   101: }
[GL4]   102: /*--------------------------------------*/
[T01]   103: /* EOF */
```

Listing 4. »ext.c«, ein MS-DOS-Filter, der aus einer Verzeichnisausgabe alle Programmdateien heraussucht

Dezimalzahlen umrechnen in GW-Basic

Oft steht man vor dem Problem, daß ein Programm oder auch der Taschenrechner eine Dezimalzahl als Ergebnis ausgibt. Viel lieber hätte man aber diese Zahl in Form eines Bruchs. Das kurze Programm »bruchd2a.bas« (Listing 5) wandelt eine Dezimalzahl in einen echten (gemischten) Bruch um. Ursprünglich war dieses nützliche Helferlein für einen Sharp-Taschenrechner gedacht, doch es tut auch auf dem PC seinen Dienst.

Das zu bewältigende Problem ist die doch recht große Rechenungenauigkeit des PC unter GW-Basic. In der nun vorliegenden Form arbeitet das Programm »bruchd2a.bas« für Zähler und Nenner kleiner 10000 exakt. Darüber hinaus werden aufgrund von internen Rundungsfehlern manchmal erwünschte Näherungswerte ausgegeben.

Sollten exaktere Ergebnisse notwendig sein, so muß in Zeile 40 für die Variable »S« ein entsprechend kleinerer Wert eingesetzt werden. Probieren geht auch hier über studieren. Zur Kontrolle wird ein relevanter Rechenfehler automatisch mit ausgegeben.

Das Programm beruht auf einem recht einfachen Prinzip: Die eingegebene Dezimalzahl wird in ihren Kehrwert verwandelt. Von diesem wird der ganzzahlige Anteil abgezogen und erneut der Kehrwert gebildet. Dies wiederholt sich so lange, bis (von Rundungsfehlern abgesehen) eine ganze Zahl übrigbleibt. Aus diesen natürlichen Zahlen läßt sich durch Erweitern, Addieren und Kehrwertbildung – sozusagen rückwärts – auf den ursprünglichen Dezimalbruch schließen.

Ein Beispiel soll dies verdeutlichen:

```
x := 0.3414634146
==> 1/x = 2.9285741429 (»2«)
==> 1/0.9285741429 = 1.076923076 (»1«)
==> 1/0.076923076 = 13.00000016 (»13«)
```

Aus den drei so ermittelten Zahlen 13, 1 und 2 wird nun der Bruch errechnet:

```
x = 1 / (2 + (1 / (1 + (1 / 13) ) ) )
= 1 / (2 + (1 / (14 / 13) ) )
= 1 / (2 + 13 / 14)
= 1 / (41 / 14)
= 14 / 41 !
```

Wie man sieht, sind nur wenige (drei) Zahlen zu berechnen, um den Bruch vollständig rekonstruieren zu können. Auch für wesentlich kompliziertere Brüche sind in der Regel kaum mehr als zehn Schritte notwendig. Die hierfür vorgesehene Anzahl der Variablen wird in Zeile 40 bereitgestellt. Sind extrem genaue Werte erforderlich und wird die Variable S verkleinert, so ist eventuell auch die Anzahl der nötigen Variablen von derzeit 25 zu erhöhen. Dies ist allerdings aufgrund der Rechenungenauigkeit auf einem PC ohne Arithmetikprozessor nicht sinnvoll, auf einem programmierbaren Taschenrechner aber gut möglich.

(Dirk Zayko/tr)

```
[DG4]  10 CLS:PRINT "BRUCHWANDLUNG"
[PF3]  20 CLEAR : DEFDBL X,Y,Z
[XD7]  30 INPUT "X= ",X : G=INT(X) : X=X-G : IF X=0 THEN
       30
[P04]  40 DIM X(25),Y(25),Z(25) : S=.0005
[DR2]  50 X(0)=1/X : GOTO 70
[X12]  60 X(A)=1/Z(A-1)
[7S2]  70 Y(A)=INT(X(A)+S)
[V52]  80 Z(A)=X(A)-Y(A)
[7P8]  90 IF ABS(X(A)-INT(X(A)+S))<S THEN X(A)=INT(X(A)+
       S) : GOTO 140
[P8A] 100 IF A>=25 THEN BEEP : BEEP : PRINT "KEINE BRUC
       HBILDUNG MOEGLICH!"
[WB7] 110 IF A>=25 THEN WHILE INKEY$="" : WEND : GOTO 3
       0
[9P2] 130 A=A+1 : GOTO 60
[TD5] 140 IF A=0 THEN Z=1 : N=X(0) : GOTO 190
[G62] 150 Z=1 : N=Y(A)
[VL3] 160 FOR B=A-1 TO 0 STEP-1
[KG4] 170 Z=Z+N*Y(B) : C=N : N=Z : Z=C
[FG1] 180 NEXT B
[451] 190 BEEP
[406] 200 PRINT " ="STR$(G)" +"STR$(Z)" /"STR$(N)
[4E9] 210 DEFSNG F : F=Z/N-X : IF ABS(F)>1E-10 THEN PRI
       NT "Fehler: ";F
[L65] 220 WHILE INKEY$="" : WEND : GOTO 20
```

Listing 5. »bruchd2a.bas« verwandelt eine Dezimalzahl in einen gemischten Bruch

Reset Off

Das folgende Utility wurde aus der Not geboren, ein Programm für eine öffentliche Vorführung möglichst absturzsicher zu gestalten. Hierzu sollte auch der Tastatur-Reset abgeschaltet werden. Eine Möglichkeit wäre gewesen, den Tastaturtreiber »Keybgr« entsprechend zu verändern. Diese Lösung erwies sich aber als eher lästig, weil dadurch zwei Versionen von »Keybgr« nötig sind, um den Tastatur-Reset wieder nutzen zu können. Die zweite Möglichkeit, die Tastatur durch eine spezielle Prozedur auszuschalten, solange keine Eingabe erwartet wird, schied ebenfalls aus. Hier hätte man nämlich statt der gewünschten Eingabe [Alt-Ctrl-Del] drücken können. Die eleganteste Lösung bietet das folgende Assemblerprogramm »resetoff.asm« (Listing 6).

Nach dem Start wird der Vektor für die Tastatur-Interrupt-Routine auf eine eigene Routine verbogen. Diese prüft nun, ob die Ctrl- und Alt-Tasten gleichzeitig gedrückt sind. Ist dies der Fall, wird weiter untersucht, ob auch die Del-Taste niedergedrückt ist. Die normale Funktion der Tastenkombination wird dadurch umgangen, daß das Programm einen Reset an den Tastatur-Controller und ein »EOI« (End of Interrupt) an den Interrupt-Controller sendet. Anschließend verzweigt das Programm in die normale Interrupt-Routine.

Die Reset-Funktion [Alt-Ctrl-Del] reaktivieren Sie durch einen erneuten Aufruf von »Resetoff«. Als Com-Datei belegt das Programm nur wenige Bytes, daher finden Sie in Listing 7 den Hexdump zum schnelleren Abtippen.

Sie können sich nun eine kleine Stapeldatei nach folgendem Vorbild erstellen:

```
resetoff
(Programmaufruf)
resetoff
```

Diese Stapeldatei schaltet den Tastatur-Reset vor dem Programmaufruf aus und danach wieder ein. Ein Abbrechen des Programms mit [Alt-Ctrl-Del] ist jetzt nicht mehr möglich. Nach Beendigung des Programms wird erneut »resetoff« aufgerufen und damit der ursprüngliche Zustand wiederhergestellt. Wer die Meldungen des Programms als störend empfindet, kann diese auch unterdrücken. Die Meldung wird vom Standardausgabegerät einfach mit

```
resetoff >nul
```

auf das nicht existente Gerät »nul« umgeleitet:

(Bodo Joest/tr)

```
[8G2]    1:  TITLE RESETOFF
[ØØØ]    2:
[ØØØ]    3:
[HC4]    4: ;START VOM SEGMENT »CODE«
[U33]    5: CSEG   SEGMENT 'CODE'
[EG4]    6: ;PROGRAMM STARTET AB 1ØØ HEX
[L55]    7: ;(FÜR COM-DATEIEN ERFORDERLICH)
[WP1]    8:        ORG       1ØØH
[AP5]    9: ;SEGMENT »CODE« IST FÜR DIE
[Z65]   1Ø: ;SEGMENTREGISTER CS, DS UND ES
[B42]   11: ;VORGESEHEN
[7V4]   12: ASSUME CS:CSEG,DS:CSEG,ES:CSEG
[ØØØ]   13:
[FG3]   14: START: JMP       INSTALL
[ØØØ]   15:
[ØØØ]   16:
[Z92]   17: ;DATEN-BEREICH
[ØØØ]   18:
[VW3]   19: ;POINTER AUF ALTEN INT 9
[UP2]   2Ø: ORG_VEK_9     DD   ?
[6B5]   21: ;KENNZEICHEN FÜR INSTALLATION
[C83]   22: TEST_FLAG     DB   'RO'
[ØØØ]   23:
[ØØØ]   24:
[ZB4]   25: ;NEUE INTERRUPT-9-ROUTINE
[ØØØ]   26:
[5W3]   27: ;NEUE TASTATUR-ROUTINE
[WU2]   28: INT9          PROC FAR
[ZB4]   29: ;INTERRUPTS AUSSCHALTEN
[EMØ]   3Ø:               CLI
[YV2]   31: ;REGISTER RETTEN
[PL1]   32:               PUSH      DS
[FL1]   33:               PUSH      AX
[W52]   34:               XOR       AX,AX
[X52]   35:               MOV       DS,AX
[YE4]   36: ;TASTATUR-STATUS EINLESEN
[UE3]   37:               MOV       AL,DS:[Ø417H]
[ZV1]   38:               AND       AL,12
[D64]   39: ;CTRL UND ALT DEGRÜCKT?
[GX1]   4Ø:               CMP       AL,12
[516]   41: ;NEIN --> NORMALE INTERRUPT-ROUTINE
[UY2]   42:               JNE       OLD_INT_9
[255]   43: ;JA   --> NÄCHSTEN WERT EINLESEN
[EU1]   44:               IN        AL,6ØH
[RY1]   45:               AND       AL,53H
[VD3]   46: ;DEL-TASTE GEDRÜCKT?
[8Ø2]   47:               CMP       AL,53H
[792]   48: ;JA   --> RESET
[N12]   49:               JE        RESET
[516]   5Ø: ;NEIN --> NORMALE INTERRUPT-ROUTINE
[ØØØ]   51:
[ØØØ]   52:
[6B7]   53: ;RÜCKSPRUNG-ROUTINE FÜR DEN INTERRUPT
[ØØØ]   54:
[484]   55: ;REGISTER ZURÜCKSETZEN
[NL3]   56: OLD_INT_9:    POP       AX
[GH1]   57:               POP       DS
[9V3]   58:               JMP       CS:[ORG_VEK_9]
[ØØØ]   59:
[ØØØ]   6Ø:
[HE5]   61: ;RESET FÜR TASTATUR-CONTROLLER
[ØØØ]   62:
[NG5]   63: ;STATUS VOM CONTROL-PORT LESEN
[KK3]   64: RESET:        IN        AL,61H
[1N2]   65: ;IN AH SPEICHERN
[P32]   66:               MOV       AH,AL
[KK1]   67: ;RESETBIT
[CV1]   68:               OR        AL,8ØH
[H42]   69: ;RESET SENDEN
[U22]   7Ø:               OUT       61H,AL
[MA4]   71: ;ALTEN STATUS WIEDER IN AL
[R22]   72:               XCHG      AH,AL
[MP4]   73: ;ALTEN STATUS AN CONTROLLER
[U22]   74:               OUT       61H,AL
[ØØØ]   75:
[ØØØ]   76:
[KB8]   77: ;END OF INTERRUPT AN INTERRUPT CONTROLLER S
            ENDEN
[ØØØ]   78:
[6R4]   79: ;END OF INTERRUPT IN AL UND
[512]   8Ø:               MOV       AL,2ØH
[YL5]   81: ;AN INTERRUPT CONTROLLER SENDEN
[922]   82:               OUT       2ØH,AL
[484]   83: ;REGISTER ZURÜCKSETZEN
[9H1]   84:               POP       AX
[GH1]   85:               POP       DS
[CD1]   86: ;RETURN
[GRØ]   87:               IRET
[ØØØ]   88:
[Z72]   89: INT9          ENDP
[ØØØ]   9Ø:
[ØØØ]   91:
[Y15]   92: ;BEGINN DES TRANSIENTEN CODES
[ØØØ]   93:
[CU3]   94: INSTALL       PROC      NEAR
[EMØ]   95:               CLI
[NF6]   96: ;ADRESSE DER INTERRUPT-ROUTINE 9 HOLEN
[322]   97:               MOV       AL,Ø9H
[S12]   98:               MOV       AH,35H
[NK1]   99:               INT       21H
[5H6]  1ØØ: ;TEST, OB "RESETOFF" BEREITS INSTALLIERT
[ZU5]  1Ø1:               MOV       AX,WORD PTR ES:[TES
           T_FLAG]
[3V5]  1Ø2:               MOV       DX,WORD PTR CS:[TES
           T_FLAG]
[D42]  1Ø3:               CMP       AX,DX
[YN2]  1Ø4:               JE        INSTALLED
[AA7]  1Ø5: ;INT 9 ORIGINALVEKTOR FÜR SPÄTER SPEICHERN
[VD8]  1Ø6:               MOV       WORD PTR CS:[ORG_VE
           K_9],BX    ;OFFSET  INT 9
[5N8]  1Ø7:               MOV       WORD PTR CS:[ORG_VE
           K_9+2],ES  ;SEGMENT INT 9
[8C4]  1Ø8: ;NEUEN INT 9 VEKTOR SETZEN
[MP3]  1Ø9:               MOV       DX,OFFSET INT9
[322]  11Ø:               MOV       AL,Ø9H
[N12]  111:               MOV       AH,25H
[NK1]  112:               INT       21H
[MA5]  113: ;INSTALLATIONSMELDUNG AUSGEBEN
[ED3]  114: ;ADRESSE DER MELDUNG
[1F4]  115:               MOV       DX,OFFSET OFF_TEXT
[5H3]  116: ;MELDUNG ÜBER INT 21H
[ET1]  117:               MOV       AH,9
[TF3]  118: ;FUNKTION 9: AUSGEBEN
[NK1]  119:               INT       21H
[YV2]  12Ø: ;PROGRAMM BEENDEN
[1C4]  121:               MOV       DX,OFFSET INSTALL
[6PØ]  122:               STI
[BC3]  123: ;UND RESIDENT MACHEN
[VK1]  124:               INT       27H
[ØØØ]  125:
[VZ4]  126: ;ROUTINE BEREITS INSTALLIERT
[ØD4]  127: ; --> INSTALLATION AUFHEBEN
[DT1]  128: INSTALLED:
[5P4]  129: ;INT 9 VEKTOR ZURÜCKSETZEN
[8V5]  13Ø:               MOV       DX,WORD PTR ES:[ORG
           _VEK_9]
[556]  131:               MOV       AX,WORD PTR ES:[ORG
           _VEK_9+2]
[X52]  132:               MOV       DS,AX
[322]  133:               MOV       AL,Ø9H
[N12]  134:               MOV       AH,25H
[NK1]  135:               INT       21H
[ØØØ]  136:
[ND3]  137: ;SEGMENT FREIMACHEN
[B22]  138:               MOV       AH,49H
[NK1]  139:               INT       21H
[KL1]  14Ø:               PUSH      CS
[GH1]  141:               POP       DS
[ØØØ]  142:
[HW5]  143: ;INSTALLATIONSMELDUNG AUSGEBEN UND
[R24]  144: ;PROGRAMM NORMAL BEENDEN
[XA4]  145:               MOV       DX, OFFSET ON_TEXT
[Z12]  146:               MOV       AH,Ø9H
[NK1]  147:               INT       21H
[6PØ]  148:               STI
[3D2]  149:               MOV       AX,4CØØH
[NK1]  15Ø:               INT       21H
[ØØØ]  151:
[CM2]  152: INSTALL       ENDP
[ØØØ]  153:
[ØØØ]  154:
[RT1]  155: ;MELDUNGEN
[NC7]  156: OFF_TEXT      DB   13,1Ø," ALT-CTRL-DEL OFF
           ",13,1Ø,"$"
[XA7]  157: ON_TEXT       DB   13,1Ø," ALT-CTRL-DEL ON
           ",13,1Ø,"$"
[ØØØ]  158:
[BUØ]  159: ;ENDE
[X62]  16Ø: CSEG          ENDS
[C72]  161:               END       START
```

Listing 6. »resetoff.asm« deaktiviert [Alt-Ctrl-Del]

```
[DF5]    1: EB 39 9Ø ØØ ØØ ØØ ØØ 52 4F FA 1E 5Ø 33 CØ 8E D8
[4D5]    2: AØ 17 Ø4 24 ØC 3C ØC 75 Ø8 E4 6Ø 24 53 3C 53 74
[YL5]    3: Ø7 58 1F 2E FF 2E Ø3 Ø1 E4 61 8A EØ ØC 8Ø E6 61
[3R5]    4: 86 EØ E6 61 BØ 2Ø E6 2Ø 58 1F CF FA BØ Ø9 B4 35
[GH5]    5: CD 21 26 A1 Ø7 Ø1 2E 8B 16 Ø7 Ø1 3B C2 74 2Ø 2E
[8D5]    6: 89 1E Ø3 Ø1 2E 8C Ø6 Ø5 Ø1 BA Ø9 Ø1 BØ Ø9 B4 25
[WM5]    7: CD 21 BA 93 Ø1 B4 Ø9 CD 21 BA 3B Ø1 FB CD 27 26
[HF5]    8: 8B 16 Ø3 Ø1 26 A1 Ø5 Ø1 8E D8 BØ Ø9 B4 25 CD 21
[CS5]    9: B4 49 CD 21 ØE 1F BA AA Ø1 B4 Ø9 CD 21 FB B8 ØØ
[RH5]   1Ø: 4C CD 21 ØD ØA 2Ø 41 4C 54 2D 43 54 52 4C 2D 44
[GF5]   11: 45 4C 2Ø 4F 46 46 2Ø ØD ØA 24 ØD ØA 2Ø 41 4C 54
[ZG5]   12: 2D 43 54 52 4C 2D 44 45 4C 2Ø 4F 4E 2Ø 2Ø ØD ØA
[Ø8Ø]   13: 24
```

Listing 7. Das Hexdump von »resetoff.com« zum schnelleren Abtippen

Impressum

Herausgeber
Christian Widuch
Leitung des Redaktionsbüros
Volker Everts (ev), Andreas Hagedorn (hg)
Chefredakteur
Martin Aschoff (ma)
Stellvertretende Chefredakteurin
Eva-Maria Hierlmeyer (hi)
Redaktion
Gabi Börger (gb), Werner Breuer (wb), Ulli Eike (ue), Alois Kneisle (al), Manfred Kohlen (mk), Thomas Röder (tr), Stefan Willmeroth (sw), Johann Zieger (zi)
Schlußredaktion
Karl-Heinz Peter (kp), Markus Stein (ms), Andrea Voß (av)
Redaktionsassistenz
Dorothea Lehner, Barbara Mooser
Koordination
Stefan Ritter
Produktionsleitung
Gerd Köberich, Helmut Skoupy
Layout
Yvonne Hendricks, Patricia Reifenhausen, Michael Grebenstein
Satz
Claudia Küllmer, Silvia Führer, Martina Siebert, Gabriela Joseph, Marcus Geppert, Heidemarie Kohlhaas, Jürgen Ludwig
Lektorat
Susanne Mias, Dagmar Wilhelm
Reprografie und Gestaltung
Manuela Eska, Margarete Schenk, Dieter Schnobl, Silvia Erbrich, Andrea Gundlach, Peter Gajewski
Werbegestaltung
Mohamed Hawa, Petra Biehl
Fotografie
Christian Heckmann, Klaus Jatho
Bildarchiv
Heike Meister
Anzeigenverkaufsleitung
Wolfgang Schnell
Anzeigenverkauf für PLZ 1 – 5
Gerlinde Rachow (0 56 51) 80 09 - 53
Wolfgang Brill (0 56 51) 80 09 - 51
Anzeigenverkauf für PLZ 6 – 8
DMV-Verlagsbüro München
Zaunkönigweg 2c, 8000 München 82,
Telefon: (0 89) 4 39 10 87, Telefax: (0 89) 4 39 10 80
Leitung: Britta Fiebig
Anzeigenverkauf: Monika Schöbel, Jens Dhein
Anzeigenverwaltung und Disposition
Andrea Giese, Karina Ehrlich, Beate Kranz
Anschrift Redaktion:
Redaktionsbüro Everts & Hagedorn
Redaktion DOS International,
Hans-Stießberger-Str. 3, 8013 Haar/München
Telefon: (0 89) 46 10 01 -0
Telefax: (0 89) 46 10 01-90
Anschrift Verlag:
DMV Daten und Medien Verlag
Widuch GmbH & Co. KG
Fuldaer Straße 6, 3440 Eschwege
Telefon: (0 56 51) 80 09-0
Telefax: (0 56 51) 80 09-33
Vertrieb
Verlagsunion Erich Pabel-Arthur Moewig KG (VPM)
Friedrich-Bergius-Straße 20
6200 Wiesbaden
Druck
Druckerei Jungfer, 3420 Herzberg
Bezugspreise
DOS EXTRA erscheint viermal jährlich.
Einzelpreis DM 18, – , 145 öS, 18, – sfr.
Abonnementpreise
Die Preise verstehen sich grundsätzlich einschließlich Porto und Verpackung.
Inland: Bezugspreis 4 Ausgaben: DM 60, –
Europäisches Ausland: Bezugspreis 4 Ausgaben: DM 70, –
Außereuropäisches Ausland: Bezugspreis 4 Ausgaben: DM 80, –
Bankverbindungen:
Postscheck Frankfurt/M: Kto.-Nr.: 23043-608
Raiffeisenbank Eschwege:
BLZ: 522 603 85, Kto.-Nr.: 245 7008
Die Abonnementbestellung kann innerhalb einer Woche nach Auftrag beim DMV-Verlag, Postfach 250, 3440 Eschwege schriftlich widerrufen werden. Zur Wahrung der Frist reicht der Poststempel. Das Abonnement verlängert sich automatisch um 4 Ausgaben, wenn es nicht mindestens 6 Wochen vor Ablauf beim Verlag schriftlich gekündigt wird.
Manuskripte, Programme:
bitte an das Redaktionsbüro Everts & Hagedorn, Redaktion DOS International, Hans-Stießberger-Straße 3, 8013 Haar/München senden.
Für unverlangt eingesandte Manuskripte und Datenträger sowie Fotos übernehmen der Verlag und die Redaktion keine Haftung. Die Zustimmung zum Abdruck wird vorausgesetzt.
Das Urheberrecht für veröffentlichte Manuskripte liegt ausschließlich beim Verlag. Nachdruck sowie Vervielfältigung oder sonstige Verwertung von Texten nur mit schriftlicher Genehmigung des Verlages. Für veröffentlichte Programme und Basteleien übernehmen wir weder Gewähr noch Haftung. Schaltungen und verwendete Bezeichnungen müssen nicht frei von gewerblichen Schutzrechten sein. Namentlich gekennzeichnete Fremdbeiträge geben nicht in jedem Fall die Meinung der Redaktion wieder.

9 x reichhaltige Software zum Superpreis

TOOLBOX-Spezial

Spezial I: Turbo Prolog
Eine Toolbox mit Routinen zur "Mustererkennung", je einem Programm für statistische Auswertungen und analytisches Integrieren und Differenzieren mathematischer Thermen, DBase-Dateien als Prolog-Wissensbasis und ein Turbo-Prolog-Tracer. Dokumentationen zu den Programmen auf der Diskette.

IBM *5 1/4 Zoll, Best.- Nr. 2801*
3 1/2 Zoll, Best.- Nr. 2802 **35,– DM***

Spezial II: PasComp – Standard Pascal Compiler
Kompletter Quellcode des Pascal-Compilers und des P-Code-Interpreters für Turbo Pascal 4.0 und CCD-Atari ST+. Dazu jeweils die lauffähigen Compilate und jetzt auch alle Texte der PASCAL-Serie PASCOMP als Dokumentation auf der Diskette!

IBM *5 1/4 Zoll, Best.- Nr. 2804*
3 1/2 Zoll, Best.- Nr. 2805 **35,– DM***
Atari ST 3 1/2 Zoll, Best.- Nr. 2803

Spezial III: Echtzeitverarbeitung
Komplettes Echtzeitsystem zu Studienzwecken in Turbo Pascal 3.0 Quellcode: Scheduler, Debugger, Utilities und Beispiele. Anpassungshinweise an andere Compiler, Programmiertechniken und Grundlagen der Echtzeitverarbeitung als Dokumentation auf der Diskette. Auch mit Turbo Pascal 4/5 kompilierbar.

IBM *5 1/4 Zoll, Best.- Nr. 2806*
3 1/2 Zoll, Best.- Nr. 2807 **35,– DM***

Spezial IV: CALC & Kurvendiskussion
Mit dem Mathe-Compiler CALC können Formeln und Funktionen zur Auswertung direkt über die Tastatur eingegeben werden. Jetzt auch Differenzieren! Lauffähiges Demonstrationsprogramm "Kurvendiskussion" mit Bildschirmmodulen für Apple II, IBM-CGA und EGA. Komplette Quellcodes in Turbo Pascal 3.0. Mit Dokumentation auf Diskette.

IBM *5 1/4 Zoll, Best.- Nr. 2808*
3 1/2 Zoll, Best.- Nr. 2809 **35,– DM***

Spezial V: STRUKTO Plus "Pascal"
Erzeugung von Strukturdateien, grafischen Struktogrammen und Programmorganisationsplänen (POP) aus Pascal-Quelltexten oder Strukturdateien. Ausgabe wahlweise auf Drucker, Bildschirm oder in Dateien, inklusive einer Fehleranalyse. Turbo Pascal 3.0 Quellcode, problemlos auch für Turbo Pascal 4/5 verwendbar. Anleitung und Dokumentation auf der Diskette.

IBM *5 1/4 Zoll, Best.- Nr. 2811*
3 1/2 Zoll, Best.- Nr. 2812 **35,– DM***

Spezial VI: Spiel & Spaß
Die besten Spiele und Knobels aus PASCAL und TOOLBOX jetzt in Farbe: Stratego, Schiebung, Inverso, der Zahlenwurm... Dazu ein kompletter Sprite-Editor für eigene Ausgestaltungen und eine Turbo Pascal "Spiele-Unit" mit vielen grundlegenden Funktionen. Alles als Turbo Pascal 4.0 Quellcode und als lauffähige Programme. Anleitungen als Datenfile auf Diskette.

IBM *5 1/4 Zoll, Best.- Nr. 2813*
3 1/2 Zoll, Best.- Nr. 2814 **35,– DM***

Spezial VII: Deutsche Silbentrennung
Tabellengesteuerte deutsche Silbentrennung mit mehr als 99% Treffsicherheit: Dokumentierte Algorithmen und Turbo Pascal 3.0 (auch 4/5) Quellcode. Ein lauffähiges Programm fügt beliebige Trennzeichen in einen ASCII-Text ein. Dazu passend ein System zum Aufbau beliebig großer Wörterbücher. Quellcode in Modula-2 und demonstrierte Anwendungen. Dokumentation auf Diskette.

IBM *5 1/4 Zoll, Best.- Nr. 2815*
3 1/2 Zoll, Best.- Nr. 2816 **35,– DM***

Die Spezial's III, IV, V und VII liefern wir auf Wunsch auch für ausgesuchte CPM/Formate und Apple II mit CPM/Karte! Genaue Informationen bietet das Merkblatt "CPM-Databox", das Sie mit ***adressiertem*** *Rückumschlag beim Softwarevertrieb des DMV-Verlages anfordern können.*

*** Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 4,– DM bzw. 6,– DM Porto/Verpackung.**
– Unverbindliche Preisempfehlung –

Bitte benutzen Sie die Bestellkarte.

DMV-Verlag · Postfach 250 · 3440 Eschwege

ConText PRO

Textverarbeitung mit Konzept

Jetzt lieferbar

ConText PRO

Schnell, handlich, flexibel
Das praxiserprobte Textprogramm für den professionellen Einsatz

Kompromißlose WYSIWYG-Darstellung im Grafikmodus:

Fett-, Kursiv- und Breitschrift "life" auf dem Bildschirm ohne Störung durch Steuerzeichen und anderen Schnickschnack

Einfache und blitzschnelle Bedienung:

Pulldown-Menüs und Shortcut-Bedienung, Hilfs-Fenster und Mausunterstützung in jeder Arbeitsphase, bei allen Gelegenheiten

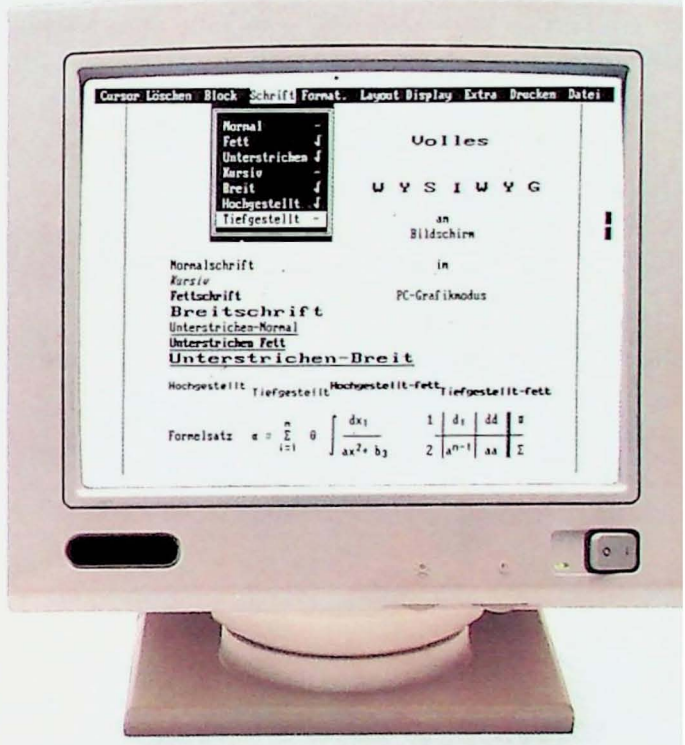
Vielseitigkeit und hohes Tempo:

alle Textverarbeitungs-Standardfunktionen plus zahlreiche unkonventionelle, durchdachte Extras für jeden Anwendungsbereich

Flexibilität:

frei konfigurierbare Tastenbelegung, Bildschirmdarstellung, Layoutgrundeinstellung, Druckeransteuerung und eigene Zeichensätze

ConText Pro nutzt die Fähigkeiten Ihres PC im Grafikmodus voll aus: Alle Schriftarten ohne störende Steuerzeichen original auf dem Bildschirm! Sie sehen genau das, was gedruckt wird, nicht mehr und nicht weniger.



Geschwindigkeit? Durch Programmierung in Assembler und mit nur 64 KByte Programmgröße ist ConText Pro auch im Grafikmodus so schnell wie eine Textausgabe: Probieren Sie es aus!

MAUSTREIBER

Für IBM PC/XT/AT und Kompatible ab 256 KByte RAM und DOS 2.0. Spezialtreiber für CGA-, EGA-, VGA- und Hercules-Grafikkarten. Unterstützung aller Zweitastenmäuse. 200 Seiten deutsches Handbuch im Ringordner.

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland DM 3,– bzw. für das Ausland DM 5,– Porto und Verpackung.
unverbindliche Preisempfehlung
Bitte benutzen Sie die Bestellkarte.

ConText Pro
Das Textprogramm für den professionellen Einsatz — 199.– DM*
5 1/4"Disk. Best.-Nr. 2407 3 1/2"Disk. Best.-Nr. 2408

ConText PC
Erprobte Version für den Einstieg — 99.– DM*
5 1/4"Disk. Best.-Nr. 228 3 1/2"Disk. Best.-Nr. 234

Upgrade auf ConText Pro — 129.– DM*
5 1/4"Disk. Best.-Nr. 2409 3 1/2"Disk. Best.-Nr. 2410

Demonstrationsdiskette — 5.– DM*
5 1/4"Disk. Best.-Nr. 0005

DMV-Verlag · Postfach 250 · 3440 Eschwege

DMV Daten- und Medienverlag

NEUE DEUTSCHE PROGRAMME

Bitte beachten Sie, daß wir über 80 ausgesuchte deutsche Shareware-programme führen. Eine komplette Liste finden Sie in unserem Diskettenkatalog.

■ PC-MS-DOS - LERNPROGRAMM — DM 12
PC Einsteiger oder Lehrende sollten sich dieses Lehrprogramm einmal genauer ansehen. Das System führt Sie schrittweise in den Umgang mit Ihrem PC ein.

■ 10FINGER - TASTATURLERNPROGRAMM — DM 20
10FINGER ist die ideale Ergänzung zu unserem PC-Lernprogramm. In 180 didaktisch aufgebauten Lektionen lernen Sie auf amüsante Weise das Blindschreiben auf Ihrem PC. Sie können Ihr persönliches Tastaturlayout anpassen. Nur im Exclusivvertrieb von Computer Solutions erhältlich.

■ POP-HELP HINTERGRUNDHILFE-DEUTSCH — DM 20
Hintergrundhilfe für alle DOS-Befehle. Ihr perfektes DOS Handbuch auf Tastendruck. Nie wieder müssen Sie mühsam im Handbuch nachschlagen - POP-HELP hilft Ihnen aus jeder Verlegenheit! Siehe auch die Beschreibung bei den englischen Programmen. Nur im Exclusivvertrieb von Computer Solutions erhältlich.

■ PC-OUTLINE - DEUTSCH — DM 12
Endlich! Die offizielle deutsche Version des wohl besten Outlining-programmes (Struktur- und Entwurfsplaner) auf dem Markt. Arbeiten auch Sie mit dem Ideenprozessor des Jahres!

■ TP 4./5.0 MULTITASKING TOOLS — DM 12
Turbo Pascal Tools zum Erstellen eigener Programme mit echtem Multitasking. So können Sie in einem Programm verschiedene Prozesse simultan ablaufen lassen. Bei Registrierung Quellcode durch den Autor erhältlich.

■ DISKETTENVERWALTUNG KAT 4.0 — DM 12
Das ultimative Disketten- und Dateiverwaltungsprogramm. Mit vielfältigen Sortier- und Selektionsmöglichkeiten und On Line Hilfe. Superprogramm! Besser als viele US-Programme.

■ ACS - ABENTEUERSPIEL - GENERATOR — DM 12
NEU! Super Adventure Creator (Quill komp.) zum Erzeugen eigener Abenteuerspiele. Mit eigener Programmiersprache, Grafikeditor, Einlesen von Fremdgrafiken, Wörter bis zu 50 Buchstaben. Für CGA oder CGA-Emulator. Klasse! Von F. Braß.

■ TRANSLATOR ENGLISCH-DEUTSCH — DM 12
NEU! Fabelhaftes Übersetzungsprogramm Englisch-Deutsch. Mit Pull-Down Menüs, Editor, On-Line Hilfe, bereits mit Wörterbuch für Computerbegriffe (beliebig erweiterbar). Ideal zum Übersetzen von Programmanleitungen. Wieder ein Top-Programm von Frank Braß. Danke!

■ Königreich unter der Erde — DM 12
Endlich einmal ein deutsches Abenteuerspiel! Amüsant und anspruchsvoll. Für Leute mit Köpfchen.

■ CASCADE 2D - CAD SYSTEM — DM 24
Neue Version des Bestsellers jetzt für CGA, EGA, VGA, HGC. Ideal für Architekten, Planer und Heimanwendungen. 2 Disks.

■ EHO-EINZELHANDEL V.2.6 — DM 12
Einzelhandelsorganisation per PC. Rechnungsschreiben, Umsatzstatistik erledigen Sie künftig per PC! Datenaustausch mit dBase mögl.

■ RW-BUCH-BUCHHALTUNG — DM 12
Die perfekte Buchhaltung für Ihren PC. Modernes System mit Pull-Down Menüs. 100 Konten, dauernde G+V. Superprogramm.

DIE BESTEN SHAREWARE PROGRAMME

■ RW-FILE - DATENBANK — DM 12
Datenbank zum schnellen und bequemen Verwalten von Daten aller Art. Mit integriertem Text- und Maskeneditor, z.B. für Serienbriefe.

■ FAKTURA - FORTE — DM 24
NEU! Superschnelles, sehr professionelles Fakturaprogramm der neuesten Generation, mit umfangreicher problembezogener Hilfefunktion, viele Selektionsmöglichkeiten, Adressenverwaltung, Etikettendruck. Für Festplatte. 2 Disks

■ VEREINSVERWALTUNG — DM 12
Komfortable Verwaltung Ihrer Vereinsdaten.

■ TURBO BASIC TOOLS — DM 12
Viele praktische Routinen für TB, z.B.: Datum, Window, Laufschrift, Eingaberoutinen, usw. Zum Einbau in eigene Programme.

■ BAUSTELLE PC — DM 69
BAUSTELLE PC ist ein sehr leistungsfähiges Programm zur Abrechnung von Baustellen im Bereich Bau/Handwerk. Für kleinere Betriebe bis 12 Mann. Abrechnung von Stunden- und Akkordlohnbaustellen, Aufmaße, Kalkulationen/Angebote/Rechnungen schreiben, Stammdaten verwalten. Festplatte nötig. 8 Disks

■ DOS-MENU — DM 12
Mit DOS-MENU fällt das Arbeiten mit dem PC leicht. Alle Programme werden bequem durch ein Menü aufgerufen.

■ Das Geheimnis der Pyramiden — DM 12
Abenteuerspiel in BASIC (mit Quellcode).

■ LERNSPIELE für Kinder — DM 12
11 Lernspiele helfen Ihren Kindern beim Lernen des Alphabets, des Einmaleins, etc...

■ VOKABELTRAINER — DM 12
Bringen Sie Ihre Fremdsprachenkenntnisse mit mit unserem Vokabeltrainer wieder auf Vordermann.

■ COMPUTER HARTETEST-GTEST — DM 12
Neue PC's sollten zu Beginn einem Dauertest unterzogen werden, wer möchte schon nach Ende der Garantiezeit der Dumme sein! GTEST entlarvt schlechte Qualität!

Goldene Regeln des Sharewarekaufs:

- ✓ ***Die Computer Solutions Library (CSL) garantiert immer brandneue Versionen der besten Sharewareprogramme***
- ✓ ***CSL Shareware wird nur auf Hochqualitätsdisketten geliefert.***
- ✓ ***Durch Direktkauf bei uns sind Sie sicher vor Computerviren! Die CSL bekommt die Programme direkt von den Autoren und nicht aus dubiosen Quellen mit Virengefahr!***

MAXI Disk

DER DURCHBRUCH BEI DOS-DISKETTENKAPAZITÄT!

Exclusiv von Computer Solutions München gibt es jetzt ein Programm, mit dem die Kapazität von MS/PC-DOS Disketten drastisch gesteigert werden kann. Folgende Steigerungen sind möglich:

420 KB	*statt*	*360 KB*
800 KB	*statt*	*720 KB*
1,4 MB	*statt*	*1,2 MB*
1,6 MB	*statt*	*1,44 MB*

Haben Sie auch genug von der sehr bescheidenen Speicherkapazität der MS/PC-DOS Disketten? Mit MAXI-Disk kennen Sie ab sofort keine Platzprobleme mehr.

MAXI-Disk erlaubt es, auf ganz normalen PC's mit Diskettenkapazitäten zu arbeiten, die bisher für unmöglich gehalten wurden (siehe oben). Dabei sind selbstverständlich keinerlei Hardwareänderung oder Spezialdisketten nötig. Einfach normale Floppy mit MAXI-Disk formatieren - fertig ist die Diskette mit viel, viel Speicherplatz. MAXI-Disk arbeitet vollkommen transparent und problemlos.

MAXI-Disk ist menügesteuert und in Deutsch. Auf vielen Rechnern lassen sich sogar Systemdisketten der erhöhten Kapazität erstellen. Dies ist besonders für Laptops oder reine Disketten-PC's interessant. Endlich hat der permanente Platznotstand ein Ende! Mehr läßt sich aus DOS-Disketten wirklich nicht herausholen!

MAXI-Disk kostet nur DM 49.- ❐
Für alle IBM-PC und 100% Kompatible ab DOS Version 3.2, auch DOS 4.0.
MAXI - Disk ist kein Sharewareprogramm

NEUE PROGRAMMPAKETE

Die beliebten **Computer Solutions Programmpakete** mit ausgesuchten Disks zu vielen Themengebieten sind eine ausgezeichnete Gelegenheit, sich mit **PROFESSIONELLER SHAREWARE zum Superpreis** zu versorgen. Besonders unsere exclusiven Programmsets für **Turbo Pascal, Assembler, BASIC, C, dBase III Clipper** und **Foxbase+** sind ein Geheimtip nicht nur bei Insidern! Schreiben auch Sie ohne große Anstrengung professionelle Programme. ***Achtung:*** Die Pakete bestehen **immer** aus jeweils 10 Disketten, falls nicht anders angegeben

■ dBase III/CLIPPER Prof. Tools — DM 299
Viele neue Disketten voll professionell einsetzbarer dBase III Programme und Tools. Inhalt: Z.B. Interrupts, Assembler, C und dBase III, Programmgeneratoren, Adressenverwaltung, dBIII Dokumentator, Tools, Tools, Tools, usw. Kurz die SUUUPERTOOLBOX für alle dBase III/Clipper/Foxbase+ Profis und solche die es werden wollen. Warum denn das Rad ständig neu erfinden? Nutzen Sie die Tools der amerikanischen dBase Profis! 50 Disketten!

■ TURBO BASIC Paket — DM 89
SIE IST DA! Die Toolbox für **echte** TB Programmierer gewartet haben. Inhalt: $INLINE, B747 Simulator, Fractals, Windows, POP UP's, Mandelbrot, Box-Drawing, usw.

■ ASSEMBLER Programmierung — DM 89
CHASM Assembler, neuer, interaktiver Disassembler, Assembler Tutorial, Assembler/Pascal Tutor, Assembler Tools I + II, MAX Freeware Editor, Wizzard Screen Maskengenerator, Assembler Utilities I + II. Viel **Assembler Quellcode!**

■ KI/PROLOG/LISP Paket — DM 89
KI-Paket mit vielen Programmen für Turbo PROLOG (u.a. ein Expertensystem im Quellcode), PD-LISP, PROLOG Interpreter, Expertensysteme. Das Power-Pack für **echte** KI-FREAKS!

■ C Programmierpakete speziell für TC u. MSC
C Routinen (Source und Libraries) für fast alle Einsatzgebiete: Windowtechnik, Serieller Port, C-Tutor, TP zu C-Übersetzer, Grafik, Sound, Joystick, Maus, Multitasking, Dateimanagement, DOS, uvm! Diese Pakete enthalten sämtliche verfügbaren C Shareware-Tools mit mehreren hundert Routinen für alle Einsatzgebiete. **Echte C Programmierer haben dieses Paket!**
WICHTIG: Unbedingt nur gewünschte Version ankreuzen:
Turbo C Profipaket mit 30 Disks nur DM 189.- ❐
Microsoft C Profipaket 30 Disks nur DM 189.- ❐

■ Turbo Pascal 4/5.0 Profipaket 30 Disks — DM189
Nachschub für **echte** Turbo Pascal 4/5.0 Programmierer! Mit: Litecomm Communications Tools, PXL das beste Crossreference Programm, Turbo Technojocks Toolkit, Turbo Overdrive Package, TP zu TC Translator mit Source, PIBTERM, EZ-DOSS Manager, Datei(de)kompression (ARC), TP Tutor, Boosters, TP4MENU und den TP 4/5.0 Multitasking Tools. Sourcecode und Units. Wer Turbo Pascal wirklich beherrschen will braucht dieses Paket, es macht sich ganz von allein bezahlt!
Unsere Turbo PASCAL 4/5.0 Sammlung wächst ständig!

■ Microsoft-QUICKBASIC Tools — DM 59
Eine Sammlung nützlicher Routinen und Anwendungen für Ihren Quickbasic Compiler, u.a. einen Fullscreen-Editor mit Source! Sehr umfangreiche Funktionsbibliotheken für Windowing, String, Mouse, usw. Ein Muß für jeden QB-Programmierer!. 6 Disks

■ Musik und Komposition — DM 89
Verwandeln Sie Ihren PC in einen Kompositionscomputer! Die besten Musikprogramme von Bach bis Mozart. Entdecken Sie völlig neue Soundfähigkeiten und die Musikalität Ihres Computers!

■ Profipaket — DM 89
zum Kennenlernen. Mit einer Auswahl der besten Programme zur Textverarbeitung, Datenbank, Datenfernübertragung, Tabellenkalkulation, DOS-Hilfsprogrammen und - zur Erholung, unseren TOP Spielen. Weiterhin sind enthalten ein Zeichen- und Malprogramm sowie ein ausgezeichneter Computer Tutor. Der ideale Einstieg!

■ MAGNUM Tools II - 10 Disks — DM 69
Hier finden Sie: Extended Batch Language zum echten Programmieren in Batch-Dateien, Newkey, BAKERS DOZEN, Jim Buttons beste Dienstprogramme, PC-PROMPT, DISK KOMMANDO, AUTOMENU dem besten DOS-Menüsystem, DISK PRINT SPOOLER mit unbegrenzter Spoolkapazität und PACKDISK. Achten Sie bitte auf die Einzelbeschreibungen. Ein echtes Muß!

SHAREWARE IST AUCH IDEAL FÜR SIE!

■ Deutsche Programme Pak II (20 Disks) — DM 149
Paket II mit unseren beliebten deutschen Programmen. Astrologie, Datenbank mit Editor, Vereinsverwaltung, Tastaturmakros, Datenfernübertragung, Großformatdruck, Buchhaltung, Adressenverwaltung, Dokuexpert, RW-FILE, EHO, PC-WINDOW, (resident) ASCII Tabelle, Editor, u.v.m; Kapitalrechnungen, Etikett, Steuer87, FIX Faktura das schnelle Programm zum Rechnungschreiben. 20 deutsche Programme zu Super-Sparpreis!

■ Deutsche Programme Pak III (50 Disks) — DM 299
Paket III mit 50 Disketten unserer beliebten deutschen Programme, zum Super-Sparpreis: Mit dem Inhalt von Paket II und zusätzlich die neuesten und besten deutschen Programme. 50 Disketten. Deutschlands Programmierer holen auf!

■ Datenfernübertragung (25 Disks) — DM 169
Die besten DFÜ-Programme: Neue Auflage unseres DFÜ-Hammers mit: QMODEM, PROCOMM, TELIX, BOYAN03, RBBS PC Mailbox, Magpie, GT Powercomm, KERMIT, ZCOMM, und DFÜ Tools! Ein außergewöhnliches Paket! 25 Disks.

■ LOTUS 1-2-3 Worksheets,Tools — DM 149
Power Worksheets! Statistik, Grafik, Datenverwaltung und Finanzen. Tips und Tricks. Werden auch Sie mit unseren vorgefertigten Arbeitsblättern 1-2-3 zum Lotus 1-2-3 Profi! 20 Disks
Wir empfehlen hierzu unseren Lotus 1-2-3 Clone AsEasyAs, das Superspreadsheet!

■ SYMPHONY Arbeitsblätter und Tools — DM 149
Entdecken Sie mit unseren Symphony Tools und Spreadsheets die volle Leistungsfähigkeit dieses Superprogrammes. Die 20 Disketten sind eine Sinfonie in Symphony, die ihresgleichen sucht! Themen ähnlich unserem 1-2-3 Paket. Die Symphony Tech Notes sind alleine schon das Geld wert! Dieses Paket sollte wirklich jeder Symphony Anwender sein Eigen nennen. Ideal auch für Firmeneinsatz, da viel Programmierarbeit gespart wird. 20 Disks

■ HP-LASERJET-DTP-TOOLKIT 30 Disks — DM 189
30 Disketten-Set für den Laserjetanwender. Dieses Toolkit beinhaltet: 5 nützliche Utilitydisketten (z.B. res. Printer Control, Barcodedruck, Screen Capture, WP Macros, usw.) Dazu: Fonts für MS-WORD. Für Ventura Publisher 6 Disketten Fonts von 8 bis 60 Pt Größe in verschiedenen Schnitarten! GEM Clipart für Ventura und GEM. Postscript- und BIT mapped Zeichenprogramme mit GEM IMG Export. Laserjet Besitzer bitte zugreifen!

■ EGA-GAMES — DM 89
Die besten EGA-Games mit Super-Grafik: EGA- TREK, -RISK, Blackjack, uvm... Alles in Super EGA-Grafik! 10 Disks

■ Supergames für CGA/CGA-Emulator — DM 89
Die besten 25 Actionspiele (Grafik) sorgen für die nötige Kurzweil an langen Winterabenden oder im Büro. Die meisten Spiele laufen auch mit unserem CGA Emulator auf der Herculeskarte.

DIE PROFESSIONELLE SHAREWARE

■ PC GOLDEN TOOLS- 10 Disks — DM 69
10er Set mit den besten MS-DOS Utilites zum Retten gelöschter Dateien, Laufwerkstestprogramm, **sehr nützlich im Datenschutz: Programme zum Verwalten versteckter Subdirectories**; außerdem Super DOS-Alarmuhr mit **automatischem Starten von Programmen**, residenter Kommandozeileneditor (ein Muß!), residenter DOS-Fixer, Tastatur Macroprozessor, die **BATCH Utilites** zeigen Ihnen die verborgenen Fähigkeiten des DOS Batch Kommados. Nicht nur für Anfänger ist das TSHELL Menüsystem und die DOS-Online Help gedacht. **Die PC GOLDEN- TOOLS sollte jeder PC Anwender sein Eigen nennen! 10 Disks**

■ DRUCKEN/TEXT Utilities — DM 89
Drucken Sie in Korrespondenzqualität und/oder (17 anderen Schriften) auf Ihrem alten Matrixdrucker, mehrspaltig oder in RIESENLETTERN. COLLIST ermöglicht Mehrspaltendruck, Iconmaker erzeugt Grafiken, Sidewriter dient zum Querdruck von breiten Texten oder Tabellen (ähnlich Sideways), raffinierte Dateifilterprogramme runden das Paket ab

TABELLENKALKULATION

■ AsEasyAs - Tabellenkalkulation — DM 12
Durch den direkten Kontakt mit dem Autor können wir wieder anbieten: Das phantastische AsEasyAs, die Lotus 123 kompatible Tabellenkalkulation mit vielen Erweiterungen, Grafik, Makros (Recorder), integrierter linearer Optimierung, Matrizen, 3-D, usw. Für CGA, EGA, Hercules, M24. Der **SUPER** Clone des Jahres. Unglaublich, was dieses Programm leistet! Über 54 Hilfebildschirme. AsEasyAs setzt sich immer mehr durch! Die mit Abstand beste SHAREWARE-Kalkulation. BESTSELLER!

EINZELPROGRAMME

PROGRAMMIERUNG

■ PERSONAL C - COMPILER — DM 12
BRANDNEU! Endlich ist er da! Der ideale C Compiler nicht nur für Einsteiger. Günstiger können Sie die C Programmierung nicht beginnen

■ MODULA2-COMPILER V.2.0 — DM 39
NEUE VERSION 2.0! Vollwertiger Modula 2 Compiler mit integrierter Entwicklungsumgebung, Editor, Make, Inline Assembler, Microsoft Link Format. Siehe Test PC Magazin. Ein Supercompiler. **Jetzt mit deutscher Anleitung!** 2 Disks

■ MODULA2-TUTOR — DM 24
Ausgezeichneter Modula2 Tutor mit vielen Beispielprogrammen. Ideal zum Erlernen von Modula2! 2 Disks

C-TUTOR — DM 24
Ausgezeichneter C-Tutor mit vielen Beispielprogrammen. Ideal zum Erlernen von C! 2 Disks
Gew. Version: Turbo C ❐ MSC ❐

■ A86/D86 — DM 32
Vollwertiger Macroassembler mit Debugger. "The best Assembler available" wird A86 in den USA genannt. **Jetzt mit deutscher Anleitung!**

DIENSTPROGRAMME

■ VIRUSKILLER-neu jetzt 2 Disks — DM 24
Sorgen Sie für wirksamen Impfschutz gegen Computerviren. Die 18 absolut besten Virenkiller aus den USA halten Ihren Computer gesund. Besser können Sie sich nicht schützen! Der Bestseller!

■ MS-OS/2 Utilities — DM 12
Computer Solutions bringt die OS/2 Utilites: Errorlevel (VIO), SuperDir, FullScreen File Lister, SUPERDEL, Kill Process, etc.

■ MS-WINDOWS PROGRAMME — DM 24
10 Top SHAREWARE-Programme für MS-WINDOWS, u.a. Filelister, etc. Ein Muß für WINDOWS Anwender. 2 Disks.

■ PC-VAULT-FESTPLATTENSCHUTZ-TEST — GRATIS
Das Super-Paßwortschutzprogramm für alle Festplatten (auch über 32MB oder unter DOS 4.0, Speedstor, etc.). Bis zu vier User- und ein Verwalterpaßwort. PC-VAULT verhindert den Unberechtigten Zugriff auf Festplatte C oder D. Reine Softwarelösung bestens auch für Laptops geeignet. Das voll professionelle Nachfolgeprodukt von PC-LOCK, exclusiv von uns. Die vollfunktionsfähige Demo mit Paßwortlänge 1 überzeugt Sie sicher. Alles in Deutsch! Ausführlicher Test in micro 3/89.

■ PROFESSIONELLE DATEIKOMPRESSION — DM 60
ÜBER 100 MB AUF EINER 20MB PLATTE. Weltweit die besten Komprimier und ARCvierprogramme. Auf 50% komprimierte EXE-Dateien bleiben lauffähig! Komprimieren ganzer Verzeichnisse oder Platten! Fantastisch! Verdoppelt die Kapazität Ihrer Festplatte. Bessere Komprimierprogramme gibt es nicht! Passwortschutz. Nur neueste Versionen. 5 komprimierte Disks.

■ POP-TRAIN - Hintergrundhilfeprogramm — DM 12
NEU! Residente On-Line Hilfe a la NORTON GUIDES für jedes Programm! Änderbare Hilfetexte werden abhängig von der Cursorposition angezeigt. POP-TRAIN sollte wirklich jeder PC-Anwender haben.

■ CONTROL-ALT NOTTASTE — DM 12
Erspart das lästige Neustarten des PC nach Systemabstürzen. Fast jedes Programm kann nach Belieben flashabgebrochen werden, ohne Schaden anzurichten. Cut- und Paste Funktion zum Datenexport. Insgesamt 40 Funktionen. Speicherresident.

■ SUPER SUCH & LIST TOOLS — DM 24
Die 19 besten SUCH & LIST Programme, wie LIST64, LOOKFOR, MAXFIND, PSEARCH, etc. Logische Suchoperationen wie, suche: **"Schulz UND HAMBURG ODER MAIER "** sind möglich. Jetzt finden Sie ALLES auf der Platte! Um Klassen besser als alle kommerziellen Programme - Garantiert! Die unentbehrlichsten Programme! 2 Disks

■ VMIX-MULTITASKING — DM 12
Super Multitaskingzusatz für DOS, mit Anlehnung an UNIX-Features. Programmierschnittstelle. Siehe Test in CHIP TOOL.

■ ALT-TASK SWITCHER — DM 12
Erlaubt das "fliegende" Umschalten zwischen zwei Anwendungen ohne die Programme abschließen zu müssen. Sie verwandeln Ihren PC in zwei PC's in einem. Unglaubliche Zeitersparnis möglich. Mit integrierter Adresskartei, Rechner, Terminkalender, Datentransfer (Cut&Paste), Tastaturmacros. Das Software-Carousel des kleinen Mannes. Ein echtes Superprogramm. (Kein Multitasking)

■ HERCULES UTILITIES — DM 12
Die besten Dienstprogramme rund um Ihre Herculeskarte. Mit 5 verschiedenen CGA Emulatoren, davon einer für Spiele mit eigenem Betriebssystem (bootresistent)! Endlich laufen viele Grafikprogramme auch auf Ihrer Herculeskarte. GWBASIC, Turbo Pascal, Spiele. Wenn Sie eine Herculeskarte haben, brauchen Sie diese unentbehrliche Diskette. Allein die Hercules Spezialtools sind ihr Geld wert!

■ SYSTEM SETUP TOOLS — DM 12
Reichhaltige Utilitydisk mit: AT und Turbo PC-Bremser, Silencer, Beliebiges Abändern des System Beeps, CMOS-Tools, Memoryset. Für PC und AT.

■ FLASHBAK - FESTPLATTENBACKUP — DM 20
das professionelle Festplattenbackupprogramm zum bequemen Sichern Ihrer wertvollen Daten und Programme, mit Pull-Down Menüs, automatischem Formatieren der Sicherungsdisketten. Unentbehrlich für alle Festplattenbesitzer. Mit vielen Optionen.
Mit deutscher Anleitung!

■ PACKDISK - DISKOPTIMIZER — DM 12
Der Diskoptimizer bringt Ihre erlahmte Festplatte wieder auf Touren! Sehr empfehlenswert, mit DELDIR und TRANSDIR, zum bequemen Löschen und Verschieben ganzer Directories!

■ PC-PROMPT - Residente DOS-Hilfe — DM 12
Vollautomatische Hilfe bei der Eingabe von Betriebssystembefehlen, innerhalb von BASIC, Turbo Pascal und DEBUG. Nicht nur für Einsteiger, sondern auch für Profis!

■ TREEVIEW und AUTOMENU — DM 24
Die absolut leistungsfähigsten Menüsysteme für Ihren PC. Wer verwendet da noch XTREE? AUTOMENU NEUE VERSION 4.5 mit Passwordschutz, Maussteuerung und vielen Extras, die es zum anerkannt besten Menüsystem machen.

■ NEWKEY - Tastaturmakros — DM 32
NEUE VERSION 5.1. Jetzt noch besser, z.B. mit Displaymacros! Belegen Sie mit NEWKEY beliebige Tasten mit beliebig langen Texten oder Befehlen, wie Floskeln, Absender, Befehle, usw. Sehr empfehlenswert! Ein absolutes Muß, vom PC-MAGAZIN in den USA ausgezeichnet! Siehe auch den Test im PC-Magazin plus. **BESTSELLER! Mit deutscher Anleitung!**

■ Extended Batch Language — DM 12
Ermöglicht echtes Programmieren innerhalb von Stapeldateien. Großrechnerkomfort mit PC-Batch Dateien.

■ PROF. MASTER KEY DISK TOOLS — DM 12
enthalten viele von den NORTON Utilites Advanced Edition her bekannten Funktionen, wie UNDELETE, UNFORMAT, Diskeditor, Attributes, usw.. Warum hunderte DM ausgeben, wenn's auch so geht!

■ Harddisktest — DM 12
Testen Sie Ihre Festplatte auf mögliche Schäden, bevor es zu spät ist! Mögliche Defekte werden frühzeitig erkannt und umgangen.

■ CP/M-2.2 -Emulator/Fremdformate — DM 12
22DISK/NICE Programmsystem zum Emulieren eines CP/M 2.2 Rechners inclusive **Lesen und Schreiben von CP/M Fremdformaten und diverser Terminalemulationen. Neue Formate** können angepaßt werden. Arbeitet mit oder ohne NEC V-xx Chips, emuliert den Z-80. DER BESTE CP/M EMULATOR!

■ PC-SPRACHAUSGABE-PROGRAMME — DM 20
Sprachausgabe für PC's. Ohne Hardwarezusätze kann Ihr PC beliebig lange Texte, in Deutsch, durch den PC Lautsprecher sprechen! Schnittstellen zu BASIC, Turbo Pascal und Assembler.

■ SPRACH- u. MUSIK Digitizer — DM 12
Beschreibung (Schaltplan fertig zum Ausdrucken auf Diskette) eines kompletten Sprach- und Musik Digitizers für Ihren PC. Erstaunliche Tonqualität! Ihr PC als Tonband. Beispielsound schon dabei. Nur wenige Bauteile nötig. Mit Software im Quellcode.

TEXTVERARBEITUNG/DRUCKEN

■ TYPE STUDIO - VENTURA - TOOLS-POSTSCRIPT DM 24
Font Generator für Ventura Publisher und Postscript Drucker mit unzähligen Variationsmöglichkeiten, wie Outline Fonts, Shading, Rotate, uvm. DAS MUSS FÜR ALLE DTP Anwender. Nur Ventura Publisher 2 Disks.

■ GALAXY WORD - TEXTVERARBEITUNG V. 2.4 — DM 32
Extrem schnelles Textsystem der vierten Generation mit Pull-Down Menus, Macros, Spellchecker, WordStar kompatibel! Textlänge bis über 200KB im RAM. Volle Druckersteuerung, auch Laser! Ein voll professionelles einsetzbares Produkt! **Mit deutscher Anleitung!**

■ CHI-WRITER - Wissenschaftliche Texte — DM 32
Technisch wissenschaftliche Textverarbeitung mit sechs verschiedenen Schriftarten (z.B. Griechisch, Mathe, usw.) in **WYSIWYG** Darstellung! Der Text wird auf dem Bildschirm genauso angezeigt wie er später ausgedruckt wird! Das Schreiben kompliziertester Formeln wird zum Kinderspiel. Für CGA-, EGA im CGA Mode und unseren CGA Emulator auf Hercules. **Mit deutscher Anleitung!**

■ EI - Texteditor — DM 12
EI Multiwindow Editor mit beliebig vielen Dateien gleichzeitig, Macros, Automatischer Compileraufruf, mit Positionierung auf den Fehler. Ideal für Profi- Programmierer. Mit EGA/VGA Support. Spitzenprodukt!

■ BLACKBEARD - PROFIEDITOR — DM 12
BLACKBEARD der Supereditor. Superschnell. Viele Funktionen, leistungsfähige Macrosprache. **Unbegrenzte Textgröße, 9 Dateien gleichzeitig in Fenstern, residenter Modus.** Einziger Sharewareeditor mit virtueller Speichertechnik. Wer kann mehr?

COMPUTER SOLUTIONS MÜNCHEN

Computer Solutions Software GmbH
Postfach 221253 • D-8000 München 22
✆ 089/295976 FAX 089/2913703 BTX 089296793
Ladenverkauf Computershop Ottobrunn
Lauinweg 14 · 8012 Ottobrunn, Tel 089/6098639

Wir suchen noch einige Fachhändler für unsere Shareware- und kommerziellen Produkte.

■ LQ-PRINT - Qualität für 9-Nadel Drucker — DM 12
Nun können Sie mit Ihrem "alten" 9 Nadeldrucker endlich auch in Briefqualität drucken. Und das auch noch in vielen verschiedenen Schriftarten. Auch für viele 24 Nadeldrucker verwendbar. Mit grafischem Briefkopf Generator und Plakatdruck. SUPER! BESTSELLER!

DATENBANKEN

■ WAMPUM - Datenbank im dB III Format — DM 24
Extrem schnelle dBase III datenkompatible Datenbank. Einfache Menüsteuerung, programmierbare Serienbriefe, Editor, Makros, Maskeneditor. Eines der besten SHAREWARE- Programme. 2 Disks

■ WAMPUM - FOXBASE+ Version — DM 149
Was alle Wampum Anwender bisher vermißt haben, wird jetzt endlich wahr: Programmierbarkeit in der dBase III+ Sprache und gewohnter dBase III interaktiver Modus (ohne Assist). Alle Foxbase+ Erweiterungen stehen Ihnen zur Verfügung. WAMPUM Foxbase+ ist DIE Power-Datenbank nicht nur für den kleinen Geldbeutel! Diese Version ist glatt bis zu 6x schneller als die Datenbank, die Sie kennen! Keine Shareware, exclusiv von uns!

■ dLITE-Residente Datenbank — DM 12
NEU! dLITE ist eine speicherresidente dBase III datenkompatible Datenbank. Sofortkopierfunktion der Daten in die laufende Textverarbeitung. Ideal zur Adressensuche! Superprogramm!

DATENFERNÜBERTRAGUNG

■ PIBTERM DFÜ-PROGRAMM (mit Source) — DM 60
Super DFÜ-PROGRAMM mit Script, Hostmode, Terminalemulation (auch Grafik), Maus, Texteditor, COM1-COM4 Support, alle Übertragungsprotokolle. PIBTERM bietet mehr als viele Kommerzprodukte. Unfaßbar: **PIBTERM wird inclusive komplettem Sourcecode in Turbo Pascal 4.0 geliefert und ist gleichzeitig eine fantastische Toolbox aller Anwendungen von Turbo Pascal 4.0. Über 35000 Zeilen kommentierter TP 4.0 Quellcode vom Feinsten! 6 Disks**

■ PROCOMM (V. 2.42) — DM 12
Nach wie vor beliebt und viel im Einsatz, das gute alte Procomm mit seinen vielen Funktionen.

■ TELIX 3.1 - Telecom — DM 24
Voll professionelles DFÜ Programm mit allen Schikanen, leistungsfähige Scripts, alle Protokolle. Speed bis 115200 Baud! Schwere Zeiten nahen für PROCOMM. SUPERPROGRAMM! 2 Disks

■ WEAKLINK - schnelle PC Kommunikation — DM 12
superschnelle Übertragung zwischen zwei PC's (mit 115200 Baud). Die seriell verbundenen Rechner kommunizieren auf der normalen DOS-Ebene miteinander. Die Laufwerke des einen Rechners können vom anderen PC wie dessen eigene Disketten oder Festplatte angesprochen werden. SUPERPROGRAMM!

■ BOYAN D4-DFÜ NEUE VERSION — DM 12
Allerneueste Version des DFÜ-Superprogrammes. BOYAN wurde vom PC-MAGAZINE in den höchsten Tönen gelobt und als Vorbild eines modernen DFÜ Programmes bezeichnet. DFÜ-Experten bitte zugreifen!

■ RBBS-PC - Mailbox - NEUEVERSION 17.1C — DM 60
Das beste Mailboxprogramm aus den USA. Mit allen Schikanen. RBBS ist ausführlich dokumentiert und der wohl beste Weg zur eigenen Mailbox! RBBS-PC ist weltweit eine der meistverbreiteten Mailboxen. Sie würden staunen, wenn Sie wüßten welche bekannten Firmen RBBS einsetzen. Leider dürfen wir es Ihnen nicht sagen. Steigen auch Sie um in den Mailbox Porsche zum Matchbox-Preis! 5 Disks.

UNTERHALTUNG

■ 3D-CHESS - 3-D Schachprogramm — DM 12
Das Profi-Schachprogramm für Anspruchsvolle. Mit voller 3-D Grafikdarstellung aller Figuren und Spielzüge. Können Sie es schlagen? Für CGA und Hercules.

■ COMPOSER - Kompositionscomputer — DM 12
Mit unserem Composer ist es möglich, auf grafischen Notenblättern Musikstücke zu komponieren, abzuspielen, abzuspeichern, editieren und auch auszudrucken. Für CGA und unseren CGA-Emulator.

GRAFIK UND CAD

■ EPG - Easy Presentation Graphics — DM 36
Brandheiß: Das voll professionelle Grafik-Präsentationssystem mit wirklich allen Schikanen. Vergleichen Sie EPG mit den besten kommerziellen Systemen und Sie werden von EPG begeistert sein. 12 Grafikgrundtypen, freier Text, 3-D Effekte, Pictogramme, etc. Über 225 Drucker werden unterstützt. Superdruckqualität auch auf 9-Nadeldruckern (240 dpi, Laser 300dpi.) (Sharewareversion mit BrownBag Logo im Ausdruck). Für CGA/EGA/VGA/VEGA7/Hercules, Laser (HP)/9/24-Pindrucker, Plotter. BESTSELLER! **Neu 3 Disks**

■ DANCAD3D - 3-DIMENSIONALES CAD — DM 48
Exklusiv von uns: Echtes dreidimensionales CAD System mit Animation in Farbe. Das System ermöglicht das Erstellen von 3-D Grafiken mit Animation. Der Kopf glaubt nicht, was die Augen sehen! NEU für **CGA/EGA/Hercules. Laser-(HP- und Postscript)/24-Pindrucker Plotter. BESTSELLER! Neu 4 Disks**

■ PC-KEY-DRAW - CAD Programm — DM 48
Brandneue Version dieses ausgezeichneten PC - CAD Systemes. Mit Hyperdraw (ähnlich Macintosh). Die erstellten Grafiken können auf einem Matrixdrucker ausgegeben werden. Eines der besten Grafikprogramme! Mit vielen Beispielen. Für CGA (auch Emulator) und EGA. 4 Disks

■ DROEGE-Platinenlayout — DM 24
Ausgezeichnetes Platinenlayoutprogramm für alle Elektroniker. Für CGA. 2 Disks.

■ PC-DRAFT-III-CAD-Programm — DM 24
NEU JETZT 2 DISKS. PC-DRAFT-III* ist der neue Champion der SHAREWARE CAD Programme. Ideal für alle anspruchsvollen PC Grafikanwender. Alle bekannten Zeichenfunktionen plus eine Menge mehr: Makrosprache, automatische Bemaßung, unlimiterte Layer, gruppierte logische Objekte, Zoom und Pan, umfangreiche Symbolbibliothek, viele Schriften, direkte Schnittstelle zu GEM und DTP Ventura Publisher. Matrix, Plotter und HP-Laserjet Support bis 300 dpi. Ein Superprogramm. (* bitte nicht mit PC-DRAFT von RHV verwechseln)

■ DRAFT CHOICE- CAD PROGRAMM — DM 12
NEU! Vollwertiges objektorientiertes CAD Programm mit allen Schikanen, wie automatische Bemaßung, Flächenberechnung, Bezierkurven, Macros, Symbolbibliothek, etc. Für alle CAD Aufgaben bestens geeignet. Das beste Shareware CAD-Programm!? Für CGA/EGA/Hercules. Drucker: EPSON, Laserjet und HP Plotter. Ein neues Meisterwerk der Autoren von AsEasyAs.

Bestellschein an **Computer Solutions GmbH** • PF 221253 • D-8000 München 22
(Kein Ladenverkauf) • **ANKREUZEN UND EINSENDEN!**

Liefern Sie uns per/gegen Bar ❐ Scheck ❐
Nachnahme ❐ Rechnung ❐
Lieferung gegen Rechnung nur bei Großfirmen und Institutionen. Ausland nur EUROSCHECK bar ausgestellt (nicht zur Verrechnung!) oder BAR.

IBM-PC SHAREWARE Gesamtkatalog (auf 3 Disks) mit Beschreibungen von ca. 1600 Disketten und Stichwortsuchprogramm für nur 12 DM Schutzgebühr ❐
3,5" Format 18 DM ❐

Die Schutzgebühr wird bei Diskettenkauf von über 100 DM angerechnet (auch nachher).

DER KATALOG IST DIE GÜNSTIGE MÖGLICHKEIT SICH ÜBER ALLE SHAREWAREPROGRAME FÜR IHREN PC ZU INFORMIEREN! KENNER HABEN IHN! SIE NICHT?

Senden Sie uns Ihr Gratisinfo zu ❐
Rabattstaffel: 1-9 Disks nur DM 12.- / 10-19 St. DM 10.- / 20-29 St. DM 8.- / ab 30 St. 7.-
Bei Lieferung auf 3,5 Zoll 3 DM Aufpreis/Diskette ❐

Firmenbestellungen bitte mit Firmenpapier ausführen und Bezug auf diese Anzeige (Heftname) nehmen)

Summe: ______ DM + 6 DM Vers. = ______ DM
Ihre Kundennummer bei uns: ______ (falls vorh.)
Name: ______
Firma: ______
Straße: ______
PLZ/Ort: ______
Telefon (______) ______
Datum: ____ . ____ . 1989

DOS Extra 7